Contents

Chapter 11

Nonlinear Systems and Optimization

11.1 Systems of Nonlinear Equations

11.1.1 Introduction

Many problems can be written in the generic form

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1 : n. \tag{11.1.1}$$

where f_i are given functions of n variables. In this section we consider the numerical solution of such systems, where at least one function depends nonlinearly on at least one of the variables. Such a system is called a **nonlinear system** of equations, and can be written more compactly as

$$f(x) = 0, \qquad f: \mathbf{R}^n \to \mathbf{R}^n. \tag{11.1.2}$$

Even more generally, if f is an operator acting on some function space (11.1.1) is **functional equation**. Applications where nonlinear systems arise include initial and boundary value problems for nonlinear differential equations, and nonlinear integral equations.

The problem of finding *all* solutions of equation (11.1.1) in some subregion $\mathcal{B} \subset \mathbf{R}^n$ can be a very difficult problem. Note that in $\mathbf{R}^n$ there is no efficient method like the bisection method (see Chapter 5) that can be used as a global method to get initial approximations. In general we must therefore be content with finding local solutions, to which reasonable good initial approximations are known.

A nonlinear **optimization problem** is a problem of the form

$$\min_x \phi(x), \qquad x \in \mathbf{R}^n, \tag{11.1.3}$$

where the **objective function** ϕ is a nonlinear mapping $\mathbf{R}^n \to \mathbf{R}$. Most numerical methods try to find a **local minimum** of $\phi(x)$, i.e., a point x^* such that $\phi(x^*) \leq \phi(y)$ for all y in a neighborhood of x^*. If the objective function ϕ is continuously differentiable at a point x then any local minimum point x of ϕ must satisfy

$$g(x) = \nabla\phi(x) = 0, \tag{11.1.4}$$

where $g(x)$ is the gradient vector. This shows the close relationship between solving optimization problems and nonlinear systems of equations.

Optimization problems are encountered in many applications such as operations research, control theory, chemical engineering, and all kinds of curve fitting or more general mathematical model fitting. The optimization problem (11.1.3) is said to be **unconstrained**. In this chapter we consider mainly methods for unconstrained optimization. Methods for linear programming problems will be discussed in Section 11.4.

If in (11.1.1) there are $m > n$ equation we have an overdetermined nonlinear system. A least squares solution can then be defined to be a solution to

$$\min_{x \in \mathbf{R}^n} \phi(x), \qquad \phi(x) = \tfrac{1}{2}\|f(x)\|_2^2, \tag{11.1.5}$$

which is a **nonlinear least squares problem**. Note that this is an (unconstrained) optimization problem, where the objective function ϕ has a special form. Methods for this problem are described in Section 11.2.

Frequently the solution to the optimization problem (11.1.3) is restricted to lie in a region $\mathcal{B} \subset \mathbf{R^n}$. This region is often defined by inequality and equality constraints of the form

$$c_i(x) = 0, \quad i = 1 : m_1, \quad c_i(x) \geq 0, \quad i = m_1 + 1 : m. \tag{11.1.6}$$

There may also be constraints of the form $l_i \leq c_i(x) \leq u_i$. In the simplest case the constraint functions $c_i(x)$ are linear. Any point x, which satisfies the constraints, is said to be a **feasible point** and the set $\mathcal{B}$ is called the **feasible region**. An important special case is **linear programming** problems, where both $\phi(x)$ and the constraints (11.1.6) are linear. This problem has been extensively studied and very efficient methods exist for their solution; see Section 11.4

11.1.2 Generalized Linear Methods

In Chapter 5 we developed methods for solving a single nonlinear equation $f(x) = 0$, $f : \mathbf{R} \to \mathbf{R}$. A simple way to extend these methods for solving a nonlinear system (11.1.1) is as follows.

Given approximations $x_1^{(k)}, \ldots, x_n^{(k)}$, we use the ith equation to solve for a new approaimation for x_i,

$$f_i\left(x_1^{(k)}, \ldots, x_i, \ldots, x_n^{(k)}\right) = 0, \qquad i = 1 : n, \tag{11.1.7}$$

and take the result as $x_i^{(k+1)}$. (Note that this assumes that the f_i depends on the variable x_i.) We repeat this for $k = 0, 1, 2, \ldots$, where $x^{(0)}$ is some initial approximation. The resulting method is called the **nonlinear Jacobi** method. Since (11.1.7) is a nonlinear equation in one unknown, the methods developed in Chapter 5 can be used to solve it.

An obvious variation is to use the most recent approximations for the other components and instead of (11.1.7) solve

$$f_i\left(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i, \ldots, x_n^{(k)}\right) = 0, \qquad i = 1 : n. \tag{11.1.8}$$

This gives the **nonlinear Gauss–Seidel** method. Note that in this method the *orderings of equations and variables* are important. This method can be generalized in an obvious manner to give the **nonlinear SOR** method. A great many variations are possible corresponding to which method is used to solve the secondary one-dimensional problem. In this way methods are obtained, which can be called Jacobi-Newton, Gauss–Seidel–Secant, etc. For a further discussion of such methods we refer to Ortega and Rheinboldt, [29, 1970, Section 7.4].

11.1.3 Fixed Point Iteration

In this section we generalize the theory of fixed-point iteration developed in Section 5.2 for a single nonlinear equation. Rewriting the system (11.1.1) in the form

$$x_i = g_i(x_1, x_2, \ldots, x_n), \qquad i = 1:n,$$

suggests an iterative method where, for $k = 0, 1, 2, \ldots$, we compute

$$x_i^{(k+1)} = g_i\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \qquad i = 1:n, \tag{11.1.9}$$

Using vector notations this can be written

$$x^{(k+1)} = g\big(x^{(k)}\big), \qquad k = 0, 1, 2, \ldots, \tag{11.1.10}$$

which is known as a **fixed point iteration**. Clearly, if g is continuous and $\lim_{k\to\infty} x^{(k)} = x^*$, then $x^* = g(x^*)$ and x^* solves the system $x = g(x)$. (Recall that a vector sequence is said to converge to a limit x^* if $\lim_{k\to\infty} \|x^{(k)} - x^*\| = 0$ for some norm $\|\cdot\|$, see Section 6.2.5).

Example 11.1.1.

The nonlinear system

$$\begin{aligned} x^2 - 2x - y + 1 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

defines the intersection between a circle and a parabola. The two real roots are $(1, 0)$ and $(0, 1)$. Taking $x_0 = 0.9$ and $y_0 = 0.2$ and using the following fixed point iteration

$$x_{k+1} = (y_k - 1)/(x_k - 2), \qquad y_{k+1} = 1 - x_k^2/(y_k + 1),$$

we obtain the results

k	x_k	y_k
1	0.72727273	0.32500000
2	0.53035714	0.60081085
3	0.27162323	0.82428986
4	0.10166194	0.95955731
5	0.02130426	0.99472577
6	0.00266550	0.99977246
7	0.00011392	0.99999645
8	0.00000178	0.99999999

Note that although we started close to the root $(1, 0)$ the sequence converges to the other real root $(0, 1)$. (See also Problem 1.)

We will now derive sufficient conditions for the convergence of the fixed point iteration (11.1.10). We first need a definition.

Definition 11.1.1.
A function $f(x) : \mathbf{R}^n \to \mathbf{R}^n$, is said to **Lipschitz continuous** *in an open set $D \in \mathbf{R}^n$ if there exists a constant L such that*

$$\|f(x) - f(y)\| \le L\|x - y\|, \quad \forall x, y \in D.$$

The constant L is called a **Lipschitz constant**. *If $L < 1$ then f is called a* **contraction**.

The following important theorem generalizes Theorem 5.2.2. It not only provides a solid basis for iterative numerical techniques, but also is an important tool in theoretical analysis. Note that, *the existence of a fixed point is not assumed a priori.*

Theorem 11.1.2. The Contraction Mapping Theorem.
Let $T : E \to F$, where $E = F = \mathbf{R}^n$, be an iteration function, and $S_r = \{u \mid \|u - u_0\| < r\}$ be a ball of radius r around a given starting point $u_0 \in \mathbf{R^n}$. Assume that T is a contraction mapping in S_r, i.e.,

$$u, v \in S_r \Rightarrow \|T(u) - T(v)\| \le L\|u - v\|, \tag{11.1.11}$$

where $L < 1$. Then if

$$\|u_0 - T(u_0)\| \le (1 - L)r \tag{11.1.12}$$

the equation $u = T(u)$ has a unique solution u^* in the closure $\overline{S_r} = \{u| \ \|u - u_0\| \le r\}$. This solution can be obtained by the convergent iteration process $u_{k+1} = T(u_k)$, $k = 0, 1, \ldots$, and we have the error estimate

$$\|u_k - u^*\| \le \|u_k - u_{k-1}\|\frac{L}{1 - L} \le \|u_1 - u_0\|\frac{L^k}{1 - L}. \tag{11.1.13}$$

Proof. We first prove the uniqueness. If there were two solutions u' and u'', we would get $u' - u'' = T(u') - T(u'')$ so that

$$\|u' - u''\| = \|T(u') - T(u'')\| \le L\|u' - u''\|.$$

Since $L < 1$, it follows that $\|u' - u''\| = 0$, i.e., $u' = u''$.

By (11.1.12) we have $\|u_1 - u_0\| = \|T(u_0) - u_0\| \le (1 - L)r$, and hence $u_1 \in S_r$. We now use induction to prove that $u_n \in S_r$ for $n < j$, and that

$$\|u_j - u_{j-1}\| \le L^{j-1}(1 - L)r, \qquad \|u_j - u_0\| \le (1 - L^j)r.$$

We already know that these estimates are true for $j = 1$. Using the triangle inequality and (11.1.11) we get

$$\begin{aligned}\|u_{j+1} - u_j\| &= \|T(u_j) - T(u_{j-1})\| \le L\|u_j - u_{j-1}\| \le L^j(1-L)r,\\ \|u_{j+1} - u_0\| &\le \|u_{j+1} - u_j\| + \|u_j - u_0\| \le L^j(1-L)r + (1-L^j)r\\ &= (1-L^{j+1})r.\end{aligned}$$

This proves the induction step, and it follows that the sequence $\{u_k\}_{k=0}^{\infty}$ stays in S_r. We also have for $p > 0$

$$\begin{aligned}\|u_{j+p} - u_j\| &\le \|u_{j+p} - u_{j+p-1}\| + \cdots + \|u_{j+1} - u_j\|\\ &\le (L^{j+p-1} + \ldots + L^j)(1-L)r \le L^j(1-L^p)r \le L^j r,\end{aligned}$$

and hence $\lim_{j\to\infty} \|u_{j+p} - u_j\| = 0$. The sequence $\{u_k\}_{k=0}^{\infty}$ therefore is a Cauchy sequence, and since $\mathbf{R^n}$ is complete has a limit u^*. Since $u_j \in S_r$ for all j it follows that $u^* \in \overline{S_r}$.

Finally, by (11.1.11) T is continuous, and it follows that $\lim_{k\to\infty} T(u_k) = T(u^*) = u^*$. The demonstration of the error estimates (11.1.13) is left as exercises to the reader. □

Theorem 11.1.2 holds also in a more general setting, where $T: S_r \to \mathcal{B}$, and $\mathcal{B}$ is a Banach space[1] The proof goes through with obvious modifications. In this form the theorem can be used, e.g., to prove existence and uniqueness for initial value problems for ordinary differential equations, see Section 13.2.1.

The Lipschitz constant L is a measure of the rate of convergence; at every iteration the upper bound for the norm of the error is multiplied by a factor equal to L. The existence of a Lipschitz condition is somewhat more general than a differentiability condition, which we now consider.

Definition 11.1.3.

The function $f_i(x)$, $\mathbf{R}^n \to \mathbf{R}$, is said to be continuously differentiable at a point x if the **gradient** *vector*

$$\nabla\phi(x) = \Big(\frac{\partial f_i}{\partial x_1}, \ldots, \frac{\partial f_i}{\partial x_n}\Big)^T \in \mathbf{R}^n \tag{11.1.14}$$

exists and is continuous. The vector valued function $f(x)$, $\mathbf{R}^n \to \mathbf{R}^n$, is said to be differentiable at the point x if each component $f_i(x)$ is differentiable at x. The matrix

$$J(x) = \begin{pmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_n(x)^T \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \in \mathbf{R}^{n\times n}, \tag{11.1.15}$$

is called the **Jacobian** *of f.*

[1] A Banach space is a normed vector space which is complete, i.e., every Cauchy sequence converges to a point in $\mathcal{B}$, see Dieudeonné [11, 1961].

The following theorem shows how a Lipschitz constant for $f(x)$ can be expressed in terms of the derivative $f'(x)$.

Lemma 11.1.4.
Let function $f(x)$, $\mathbf{R}^n \to \mathbf{R}^n$, be differentiable in a convex set $\mathcal{D} \subset \mathbf{R}^n$. Then $L = \max_{y\in\mathcal{D}} \|f'(y)\|$ is a Lipschitz constant for f.

Proof. Let $0 \le t \le 1$ and consider the function $g(t) = f(a + t(x - a))$, $a, x \in \mathcal{D}$. By the chain rule $g'(t) = f'\big(a + t(x - a)\big)(x - a)$ and

$$f(x) - f(a) = g(1) - g(0) = \int_0^1 g'(t)dt = \int_0^1 f'\big(a + t(x - a)\big)(x - a)dt.$$

Since $\mathcal{D}$ is convex the whole line segment between the points a and x belongs to $\mathcal{D}$. Applying the triangle inequality (remember that an integral is the limit of a sum) we obtain

$$\|f(x) - f(a)\| < \int_0^1 \|f'(a + t(x - a))\|\ \|x - a\|dt \le \max_{y\in\mathcal{D}} \|f'(y)\|\ \|x - a\|.$$

□

11.1.4 Newton-Type Methods

Newton's method for solving a single nonlinear equation $f(x) = 0$ can be derived by using Taylor's formula to get a linear approximation for f at a point. To get a quadratically convergent method for a system of nonlinear equations we must similarly use derivative information of $f(x)$.

Let x_k be the current approximation[2] and assume that $f_i(x)$ is twice differentiable at x_k. Then by Taylor's formula

$$f_i(x) = f_i(x_k) + (\nabla f_i(x_k))^T(x - x_k) + O(\|x - x_k\|^2), \qquad i = 1 : n.$$

Using the Jacobian matrix (11.1.15) the nonlinear system $f(x) = 0$ can be written

$$f(x) = f(x_k) + J(x_k)(x - x_k) + O(\|x - x_k\|^2) = 0.$$

Neglecting higher order terms we get the linear system

$$J(x_k)(x - x_k) = -f(x_k). \tag{11.1.16}$$

If $J(x_k)$ is nonsingular then (11.1.16) has a unique solution x_{k+1}, which can be expected to be a better approximation. The resulting iterative algorithm can be written

$$x_{k+1} = x_k - \big(J(x_k)\big)^{-1} f(x_k). \tag{11.1.17}$$

[2] In the following we use vector notations so that x_k will denote the kth approximation and not the kth component of x.

which is **Newton's method**. Note that, in general the inverse Jacobian matrix should not be computed. Instead the linear system (11.1.16) is solved, e.g., by Gaussian elimination. If n is very large and $J(x_k)$ sparse it may be preferable to use one of the iterative methods given in Chapter 11. Note that in this case x_k can be used as an initial approximation.

The following example illustrates the quadratic convergence of Newton's method for simple roots.

Example 11.1.2.

The nonlinear system

$$\begin{aligned} x^2 + y^2 - 4x &= 0 \\ y^2 + 2x - 2 &= 0 \end{aligned}$$

has a solution close to $x_0 = 0.5$, $y_0 = 1$. The Jacobian matrix is

$$J(x, y) = \begin{pmatrix} 2x - 4 & 2y \\ 2 & 2y \end{pmatrix},$$

and Newton's method becomes

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - J(x_k, y_k)^{-1} \begin{pmatrix} x_k^2 + y_k^2 - 4x_n \\ y_k^2 + 2x_k - 2 \end{pmatrix}.$$

We get the results:

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868893322	1.13644297217273
4	0.35424868893541	1.13644296914943

All digits are correct in the last iteration. The quadratic convergence is obvious; the number of correct digits approximately doubles in each iteration.

It is useful to have a precise measure of the asymptotic rate of convergence for a vector sequence converging to a limit point.

Definition 11.1.5.

A convergent sequence $\{x_k\}$ *with* $\lim_{k\to\infty}\{x_k\} = x^*$, *and* $x_k \neq x^*$, *is said to have* **order of convergence** *equal to* p $(p \geq 1)$, *if*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C, \tag{11.1.18}$$

where $|C| < 1$ *for* $p = 1$ *and* $|C| < \infty$, $p > 1$. C *is called the* **asymptotic error constant**. *The sequence has exact convergence order* p *if (11.1.18) holds with* $C \neq 0$. *We say the convergence is* **superlinear** *if* $C = 0$ *for some* $p \geq 1$.

Note that for finite dimensional vector sequences, the order of convergence p does not depend on the choice of norm, and that the definitions agree with those introduced for scalar sequences, see Def. 5.2.1. (More detailed discussions of convergence rates is found in Dennis and Schnabel [10, pp. 19–21], and Chapter 9 of Ortega and Rheinboldt [29].)

In order to analyze the convergence of Newton's method we need to study how well the linear model (11.1.16) approximates the equation $f(x) = 0$. The result we need is given in the lemma below.

Lemma 11.1.6.
Assume that the Jacobian matrix satisfies the Lipschitz condition

$$\|J(x) - J(y)\| \le \gamma\|x - y\|, \qquad \forall x, y \in \mathcal{D},$$

where $\mathcal{D} \subset \mathbf{R}^n$ is a convex set. Then for all $x, y \in \mathcal{D}$ it holds that

$$\|f(x) - f(y) - J(y)(x - y)\| \le \frac{\gamma}{2}\|x - y\|^2.$$

Proof. The function $g(t) = f(y + t(x - y))$, $x, y \in \mathcal{D}$ is differentiable for all $0 \le t \le 1$, and by the chain rule $g'(t) = J(y + t(x - y))(x - y)$. It follows that

$$\|g'(t) - g'(0)\| = \|\big(J(y + t(x - y)) - J(y)\big)(x - y)\| \le \gamma t\|x - y\|^2. \qquad (11.1.19)$$

Since the line segment between x and y belongs to $\mathcal{D}$

$$f(x) - f(y) - J(y)(x - y) = g(1) - g(0) - g'(0) = \int_0^1 \big(g'(t) - g'(0)\big)dt.$$

Taking norms and using (11.1.19) it follows that

$$\|f(x) - f(y) - J(y)(x - y)\| \le \int_0^1 \|g'(t) - g'(0)\|dt \le \gamma\|x - y\|^2 \int_0^1 tdt.$$

□

The following famous theorem gives rigorous conditions for the quadratic convergence of Newton's method. It also shows that Newton's method in general converges provided that x_0 is chosen sufficiently close to a solution x^*.

Theorem 11.1.7. (Newton–Kantorovich Theorem)
Let $f : \mathbf{R}^n \to \mathbf{R}^n$ be continuously differentiable in an open convex set $C \in \mathbf{R}^n$, and Let the Jacobean matrix of $f(x)$ be $J(x)$. Assume that $f(x^*) = 0$, for $x^* \in \mathbf{R}^n$. Let positive constants r, $\beta > 0$ be given such that $S_r(x^*) = \{x| \, \|x - x^*\| < r\} \subseteq C$, and

(a) $\|J(x) - J(y)\| \le \gamma\|x - y\|, \qquad \forall \, x, y \in S_r(x^*),$

(b) $J(x^*)^{-1}$ *exists and satisfies* $\|J(x^*)^{-1}\| \le \beta$.

Then there exists an $\epsilon > 0$ *such that for all* $x_0 \in S_\epsilon(x^*)$ *the sequence generated by*

$$x_{k+1} = x_k - J\big(x_k\big)^{-1} f(x_k), \quad k = 0, 1, \ldots$$

is well defined, $\lim_{n\to\infty} = x^*$*, and satisfies*

$$\|x_{k+1} - x^*\| \le \beta\gamma \|x_k - x^*\|^2.$$

Proof. We choose $\epsilon = \min\{r, 1/(2\beta\gamma)\}$. Then by (a) and (b) it follows that

$$\|J(x^*)^{-1}(J(x_0) - J(x^*))\| \le \beta\gamma\|x_0 - x^*\| \le \beta\gamma\epsilon \le 1/2.$$

By Corollary 6.6.1 and (b) we have $\|J(x_0)^{-1}\| \le \|J(x^*)^{-1}\|/(1 - 1/2) = 2\beta$. It follows that x_1 is well defined and

$$\begin{aligned} x_1 - x^* &= x_0 - x^* - J(x_0)^{-1}\big(f(x_0) - f(x^*)\big) \\ &= J(x_0)^{-1}\left(f(x^*) - f(x_0) - J(x_0)(x^* - x_0)\right). \end{aligned}$$

Taking norms we get

$$\begin{aligned} \|x_1 - x^*\| &\le \|J\big(x_0\big)^{-1}\|\, \|f(x^*) - f(x_0) - J(x_0)(x^* - x_0)\| \\ &\le 2\beta\gamma/2\|x_0 - x^*\|^2, \end{aligned}$$

which proves quadratic convergence. □

We remark that a result by Kantorovich shows quadratic convergence under weaker conditions. In particular, it is not necessary to assume the existence of a solution , or the nonsingularity of $J(x)$ at the solution. For a discussion and proof of these results we refer to Ortega and Rheinboldt [29, 1970, Ch. 12.6].

Each step of Newton's method requires the evaluation of the n^2 entries of the Jacobian matrix $J(x_k)$, and to solve the resulting linear system $n^3/3$ arithmetic operations are needed. This may be a time consuming task if n is large. In many situations it might be preferable to reevaluate $J(x_k)$ only occasionally using the same Jacobian in $m > 1$ steps,

$$J(x_p)(x_{k+1} - x_k) = -f(x_k), \quad k = p : p + m - 1. \tag{11.1.20}$$

Once we have computed the LU factorization of $J(x_p)$ the linear system can be solved in n^2 arithmetic operations. The motivation for this approach is that if either the iterates or the Jacobian matrix are not changing too rapidly $J(x_p)$ is a good approximation to $J(x_k)$. (These assumptions do not usually hold far away from the solution, and may cause divergence in cases where the unmodified algorithm converges.)

The modified Newton method can be written as a fixed point iteration with

$$g(x) = x - J(x_p)^{-1} f(x), \quad g'(x) = I - J(x_p)^{-1} J(x).$$

We have, using assumptions from Theorem 11.1.7

$$\|g'(x)\| \le \|J(x_p)^{-1}\|\,\|J(x_p) - J(x)\| \le \beta\gamma\|x_p - x\|.$$

Since $g' \neq 0$ the modified Newton method will only have *linear* rate of convergence. Also, far away from the solution the modified method may diverge in cases where Newton's method converges.

Example 11.1.3.

Consider the nonlinear system in Example 11.1.2. Using the modified Newton method with fixed Jacobian matrix evaluated at $x_1 = 0.35$ and $y_1 = 1.15$

$$J(x_1, y_1) = \begin{pmatrix} 2x_1 - 4 & 2y_1 \\ 2 & 2y_1 \end{pmatrix} = \begin{pmatrix} -3.3 & 2.3 \\ 2.0 & 2.3 \end{pmatrix}.$$

we obtain the result

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868347696	1.13644394786146
4	0.35424868892666	1.13644298069439
5	0.35424868893540	1.13644296928555
6	0.35424868893541	1.13644296915104

11.1.5 Numerical Differentiation

It has been stated at several places in this book that numerical differentiation should be avoided, when the function values are subject to irregular errors, like errors of measurement or rounding errors. Nowadays, when a typical value of the machine constant $\mathbf{u}$ is $2^{-53} \approx 10^{-16}$, the harmful effect of *rounding errors* in the context of numerical differentiation, however, should not be exaggerated. We shall see that the accuracy of the first and second derivatives is satisfactory for most purposes, *if the step size is chosen appropriately.*

With the multilinear mapping formalism, the general case of vector valued dependent and independent variables becomes almost as simple as the scalar case. Let η be a small positive number. By Taylor's formula,

$$g'(x_0)v = \frac{g(x_0 + \eta v) - g(x_0 - \eta v)}{2\eta} + R_T, \quad R_T \approx -\frac{\eta^2 g'''(x_0)v^3}{6}, \tag{11.1.21}$$

where, as above, we use v^3 as an abbreviation for the list (v, v, v) of vector arguments. The Jacobian $g'(x_0)$ is obtained by the application of this to $v = e_j$, $j = 1 : k$. If the Jacobian has a band structure, then it can be computed by means of fewer vectors v; see Problem 3. First note that, *if g is quadratic, there is no*

truncation error, and η can be chosen rather large, so the rounding error causes no trouble either.

Suppose that the rounding error of $g(x)$ is (approximately) bounded by $\epsilon\|g\|/\eta$. (The norms here are defined on a neighborhood of x_0.) The total error is therefore (approximately) bounded by

$$B(\eta) = \|g\|\frac{\epsilon}{\eta} + \|g'''\|\|v\|^3\frac{\eta^2}{6}.$$

Set $\|g\|/\|g'''\| = \xi^3$, and note that ξ measures a local length scale of the variation of the function g, (if we interpret x as a length). A good choice of η is found by straightforward optimization:

$$\min_\eta B(\eta) = (3\epsilon)^{2/3}\|g\|\|v\|/(2\xi), \quad \eta\|v\| = (3\epsilon)^{1/3}\xi. \tag{11.1.22}$$

For $\epsilon = 10^{-16}$, we should choose $\eta\|v\| = 7{\cdot}10^{-6}\xi$. The error estimate becomes $2.5{\cdot}10^{-11}\|g\|\|v\|/\xi$. In many applications this accuracy is higher than necessary. If uncentered differences are used instead of centered differences, the error becomes $O(\epsilon^{1/2})$ with optimal choice of η, while the amount of computation may be reduced by almost 50%; see Problem 1.

It may be a little cumbersome to estimate ξ by its definition, but since we need a very rough estimate only, we can replace it by some simpler measure of the length scale of $g(x)$, e.g. a rough estimate of (say) $\frac{1}{2}\|g\|/\|g'\|$.[3] Then the error estimate simplifies to $(3\epsilon)^{2/3}\|g'\|\|v\| \approx 5{\cdot}10^{-11}\|g'\|\|v\|$ for $\epsilon = 10^{-16}$. This is usually an overestimate, though not always. Recall that if g is quadratic, there is no truncation error.

The result of a similar study of the directional *second derivative* reads

$$f''(x_0)v^2 = \frac{f(x_0+\eta v) - 2f(x_0) + f(x_0-\eta v)}{\eta^2} + R_T, \tag{11.1.23}$$

$$R_T \approx -\frac{\eta^2 f^{iv}(x_0)v^4}{12},$$

$$B(\eta) = \|f\|\frac{4\epsilon}{\eta^2} + \frac{\|f^{iv}\|\,\|v\|^4\eta^2}{12},$$

$$\xi = (\|f\|/\|f^{iv}|)^{1/4} \approx (\frac{1}{3}\|f\|/\|f''\|)^{1/2},$$

$$\min_\eta B(\eta) = 2(\epsilon/3)^{1/2}\|f\|\,\|v\|^2/\xi^2 \approx \epsilon^{1/2}\|f''\|\|v\|^2, \quad \eta\|v\| = (48\epsilon)^{1/4}\xi.$$

Note that:

- if g is a cubic function, there is no truncation error, and $\eta\|v\|$ can be chosen independent of ϵ. Otherwise, for $\epsilon = 10^{-16}$, we should choose $\eta\|v\| \approx 3{\cdot}10^{-4}\xi$. The simplified error estimate becomes $2{\cdot}10^{-8}\|f''\|\|v\|^2$;

[3] The factor $\frac{1}{2}$ is a safety factor. So is the factor $\frac{1}{3}$ in the equation for ξ in the group (11.1.23).

- if $f'(x)$ is available, we can obtain $f''(x_0)v^2$ more accurately by setting $g(x) = f'(x)v$ into (11.1.21), since the value of η can then usually be chosen smaller;
- if $f(x)$ is a quadratic form, then $f''(x)$ is a constant bilinear operator and $f''(x)v^2 = f(v)$. If f is a non-homogeneous quadratic function, its affine part must be subtracted from the right hand side;
- in order to compute $f''(x_0)(u, v)$, it is sufficient to have a subroutine for $f''(x_0)v^2$, since the following formula can be used. It is easily derived by the bilinearity and symmetry of $f''(x_0)$.

$$f''(x_0)(u, v) = \frac{1}{4}\big(f''(x_0)(u + v)^2 - f''(x_0)(u - v)^2\big) \tag{11.1.24}$$

11.1.6 Derivative Free Methods

In many applications the Jacobian matrix is not available or too expensive to evaluate. Then we can use the **discretized Newton method**, where each of the derivative elements in $J(x_k)$ is discretized separately by a difference quotient. There are many different variations depending on the choice of discretization. A frequently used approximation for the jth column of $J(x)$ is the forward difference quotient

$$\frac{\partial f(x)}{\partial x_j} \approx \Delta_j f(x) \equiv \frac{f(x + h_j e_j) - f(x)}{h_j}, \quad j = 1 : n,$$

where e_j denotes the jth unit vector and $h_j > 0$ is a suitable scalar. If the resulting approximation is denoted by $J(x, D)$, then we can write

$$J(x, D) = \big(f(x + h_1 e_1) - f(x), \ldots, f(x + h_n e_n) - f(x)\big)D^{-1},$$

where $D = \operatorname{diag}(h_1, h_2, \ldots, h_n)$ is a nonsingular diagonal matrix. This shows that $J(x_k, d)$ is nonsingular if and only if the vectors

$$f(x_k + h_j e_j) - f(x_k), \quad j = 1 : n,$$

are linearly independent.

It is important that the step sizes h_j are chosen carefully. If h_j is chosen too large then the derivative approximation will have a large truncation error; if it is chosen too small then roundoff errors may be dominate (cf. numerical differentiation). As a rule of thumb one should choose h_j so that $f(x)$ and $f(x + h_j e_j)$ have roughly the first half digits in common, i.e.,

$$|h_j| \, \|\Delta_j f(x)\| \approx u^{1/2}\|f(x)\|.$$

In the discretized Newton method the vector function $f(x)$ needs to be evaluated at $n + 1$ points, including the point x_k. Hence it requires $n^2 + n$ component function evaluations per iteration. Methods which only only require $(n^2 + 3n)/2$ component function evaluations have been proposed by Brown (1966) and Brent (1973). Brent's method requires the computation of difference quotients

$$\frac{f(x + h_j q_k) - f(x)}{h_j}, \quad j = 1 : n, \quad k = j : n,$$

where $Q = (q_1, \ldots, q_n)$ is a certain orthogonal matrix determined by the method. Note that because of common subexpressions, in some applications a component function evaluation may be almost as expensive as a vector function evaluation. In such cases the original Newton method is still to be preferred. For a discussion of these methods see Moré and Cosnard [27, 1979].

If the function $f(x)$ is complicated to evaluate even the above method may be too expensive. In the methods above we obtain the next approximation x_{k+1} by a step along the direction h_k, computed by solving the linear system

$$B_k h_k = -f(x_k), \tag{11.1.25}$$

where B_k is an approximation to the Jacobian $J(x_k)$. The class of **quasi-Newton methods** can be viewed as a generalization of the secant method to functions of more than one variable. The approximate Jacobian B_{k+1} is required to satisfy the **secant equation**

$$B_{k+1} s_k = y_k \tag{11.1.26}$$

where s_k and y_k are the vectors

$$s_k = x_{k+1} - x_k, \qquad y_k = f(x_{k+1}) - f(x_k).$$

This means that B_{k+1} correctly imitates the Jacobian along the direction of change s_k. Of course many matrices satisfy this condition.

In the very successful **Broyden's method** it is further required that the difference $B_{k+1} - B_k$ has minimal Frobenius norm. It is left as an exercise to verify that these conditions lead to

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}. \tag{11.1.27}$$

This is generally referred to as Broyden's "good" updating formula. Note that $B_{k+1} - B_k$ is *a matrix of rank one*, and that $B_{k+1} p = B_k p$ for all vectors p such that $p^T(x_{k+1} - x_k) = 0$. (To generate an initial approximation B_1 we can use finite differences along the coordinate directions.)

It can be shown that Broyden's modification of Newton's method has superlinear convergence.

Theorem 11.1.8.

Let $f(x) = 0$, $f : \mathbf{R}^n \to \mathbf{R}^n$, be sufficiently smooth, and let x^* be a regular zero point of f. Let

$$x_{k+1} = x_k - B_k^{-1} f(x_k)$$

be the Newton type method where B_k is updated according to Broyden's formula (11.1.27). If x_0 is sufficiently close to x^*, and B_0 sufficiently close to $f'(x_0)$, then the sequence $\{x_k\}$ is defined and converges superlinearly to x^*, i.e.,

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \to 0, \quad n \to \infty.$$

Proof. See Dennis and Moré [9]. □

We can compute B_{k+1} from (11.1.27) in only $2n^2$ operations and no extra function evaluations. To solve (11.1.25) for the Newton direction still seems to require $O(n^3)$ operations. However, assume that a QR decomposition $B_k = Q_k R_k$ was computed in the previous step. Then we can write

$$B_{k+1} = Q_k(R_k + u_k v_k^T), \quad u_k = Q_k^T(y_k - B_k s_k), \quad v_k^T = s_k^T / s_k^T s_k.$$

We will show below that the QR decomposition of $R_k + u_k v_k^T = \bar{Q}_k \bar{R}_{k+1}$ can be computed in $O(n^2)$ operation. Then we have

$$B_{k+1} = Q_{k+1} R_{k+1}, \qquad Q_{k+1} = Q_k \bar{Q}_k.$$

We start by determining a sequence of Givens rotations $G_{j,j+1}$, $j = n-1, \ldots, 1$ such that

$$G_{1,2}^T \ldots G_{n-1,n}^T u_k = \alpha e_1, \qquad \alpha = \pm \|u_k\|_2.$$

Note that these transformations zero the last $n-1$ components of u_k from bottom up. (For details on how to compute $G_{j,j+1}$ see Section 7.4.2.) The same transformations are now applied to the R_k, and we form

$$\bar{H} = G_{1,2}^T \ldots G_{n-1,n}^T (R_k + u_k v_k^T) = H + \alpha e_1 v_k^T.$$

It is easily verified that in the product $H = G_{1,2}^T \ldots G_{n-1,n}^T R_k$ the Givens rotations will introduce extra nonzero elements only in positions $(j, j+1)$, $j = 1 : n$, so that H becomes an upper Hessenberg matrix of the form

$$H = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \quad n = 4.$$

The addition of $\alpha e_1 v_k^T$ only modifies the first row of H, and hence also $\bar{H}$ is an upper Hessenberg matrix. We now determine a sequence of Givens rotations $\bar{G}_{j,j+1}$ so that $\bar{G}_{j,j+1}$ zeros the element $\bar{h}_{j+1,j}$, $j = 1 : n-1$. Then

$$\bar{G}_{n-1,n}^T \ldots \bar{G}_{1,2}^T \bar{H} = \bar{R}_{k+1}$$

is the updated triangular factor. The orthogonal factor equals the product

$$\bar{Q}_k = G_{n-1,n} \ldots G_{1,2} \bar{G}_{1,2} \ldots \bar{G}_{n-1,n}.$$

The work needed for this update is as follows: Computing u_k takes n^2 flops. Computing $\bar{H}$ and R_k takes $4n^2$ flops and accumulating the product of $G_{j,j+1}$ and $\bar{G}_{j,j+1}$ takes $8n^2$ flops, for a total of $13n^2$ flops. It is possible to do a similar cheap update of the LU decomposition, but this may lead to stability problems.

If the Jacobian $f'(x)$ is sparse the advantages of Broyden's method is lost, since the update in general is not sparse. One possibility is then to keep the LU factors of the most recently computed sparse Jacobian and save several Broyden updates as pairs of vectors $y_k - B_k s_k$ and s_k.

11.1.7 Modifications for Global Convergence

We showed above that under certain regularity assumptions Newton's method is convergent from a sufficiently good initial approximation, i.e., under these assumptions Newton's method is **locally convergent**. However, Newton's method is not in general **globally convergent**, i.e., it does not converge from an arbitrary starting point. Far away from the root Newton's method may not behave well, e.g., it is not uncommon that the Jacobian matrix is illconditioned or even singular. This is a serious drawback since it is *much more difficult to find a good starting point in* $\mathbf{R}^n$ *than in* $\mathbf{R}$!

We now discuss techniques to modify Newton's method, which attempt to ensure **global convergence**, i.e., convergence from a large set of starting approximations. As mentioned in the introduction the solution of the nonlinear system $f(x)$ also solves the minimization problem

$$\min_x \phi(x), \qquad \phi(x) = \frac{1}{2}\|f(x)\|_2^2 = \frac{1}{2}f(x)^T f(x). \tag{11.1.28}$$

We seek modifications which will make $\|f(x)\|_2^2$ decrease at each step. We call d a **descent direction** for $\phi(x)$ if $\phi(x + \alpha d) < \phi(x)$, for all sufficiently small $\alpha > 0$. This will be the case if if the directional derivative is negative, i.e.

$$\nabla\phi(x)^T d = f(x)^T J(x) d < 0.$$

The steepest-descent direction

$$-g = -\nabla\phi(x) = -J(x)^T f(x)$$

is the direction in which $\phi(x)$ decreases most rapidly, see Section 11.3.2.

Assuming that $J(x)$ is nonsingular, the Newton direction $h = -J(x)^{-1}f(x)$ is also a descent direction if $f(x) \neq 0$ since

$$\nabla\phi(x)^T h = -f(x)^T J(x) J(x)^{-1} f(x) = -\|f(x)\|_2^2 < 0.$$

In the **damped Newton** method we take

$$x_{k+1} = x_k + \alpha_k h_k, \qquad J(x_k)h_k = -f(x_k). \tag{11.1.29}$$

where the step length α_k is computed by a **line search**. Ideally α_k should be chosen to be minimize the scalar function

$$\psi(\alpha) = \phi\|f(x_k + \alpha h_k)\|_2^2.$$

Algorithms for solving such an one-dimensional minimization are discussed in Section 6.7. In practice this problem need not be solved accurately. It is only necessary to ensure that the reduction $\|f(x_k)\|_2^2 - \|f(x_{k+1})\|_2^2$ is sufficiently large. In the **Armijo-Goldstein criterion** α_k is taken to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \ldots$ for which

$$\psi(0) - \psi(\alpha_k) \geq \frac{1}{2}\alpha_k \psi(0)$$

is satisfied. Close to a simple zero x^* this criterion will automatically chose $\alpha_k = 1$. It then becomes identical to Newton's method and convergence becomes quadratic. Another common choice is to require that α_k satisfies the two conditions

$$\psi(\alpha_k) \le \psi(0) + \mu\alpha_k\psi'(0), \quad |\psi'(\alpha_k)| \le \eta|\psi'(0)|$$

where typically $\mu = 0.001$ and $\eta = 0.9$. The first condition ensures a sufficient decrease in $\|f(x)\|_2^2$ and the second that the gradient is decreased by a significant amount.

The addition of line searches to the Newton iteration greatly increases the range of nonlinear equations that can successfully be solved. However, if the Jacobian $J(x_k)$ is nearly singular, then h_k determined by (11.1.29) will be large and the linear model

$$f(x_k + \alpha_k h_k) \approx f(x_k) + \alpha_k J(x_k)h_k$$

inadequate. In this case the Newton direction tends to be very inefficient.

The idea in **trust region methods** is to avoid using a linear model outside its range of validity, see also Section 11.2.3. Here one takes $x_{k+1} = x_k + d_k$, where d_k solves the constrained linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2, \text{ subject to } \|d_k\|_2 \le \Delta_k,$$

where Δ_k is a parameter, which is updated recursively. If the constraint is binding this problem can be solved by introducing a Lagrange parameter λ and minimizing

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2 + \lambda\|d_k\|_2^2. \tag{11.1.30}$$

Here λ is determined by the **secular equation** $\|d_k(\lambda)\|_2 = \Delta_k$. Note that the problem (11.1.30) is equivalent to the linear least squares problem

$$\min_{d_k} \left\| \begin{pmatrix} f(x_k) \\ 0 \end{pmatrix} + \begin{pmatrix} J(x_k) \\ \lambda^{1/2} I \end{pmatrix} d_k \right\|_2^2,$$

where the matrix always has full column rank for $\lambda > 0$.

A typical rule for updating Δ_{k+1} is to first calculate the ratio ρ_k of $\|f(x_k)\|_2^2 - \|f(x_k+d_k)\|_2^2$ to the reduction $\|f(x_k)\|_2^2 - \|f(x_k)+J(x_k)d_k\|_2^2$ predicted by the linear model. Then we take

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\|d_k\|, & \text{if } \rho_k \le 0.1; \\ \Delta_k, & \text{if } 0.1 < \rho_k \le 0.7; \\ \max\{2\|d_k\|, \Delta_k\}, & \text{if } \rho_k > 0.7. \end{cases}$$

The trust region is made smaller if the model is unsuccessful and is increased if a substantial reduction in the objective function is found. A difference to line search methods is that if $\Delta_{k+1} < \Delta_k$ we set $x_{k+1} = x_k$.

A related idea is used in **Powell's hybrid method**, where a linear combination of the steepest descent and the Newton (or the quasi-Newton) direction is used. Powell takes

$$x_{k+1} = x_k + \beta_k d_k + (1 - \beta_k)h_k, \quad 0 \le \beta_k \le 1,$$

where h_k is the Newton direction in (11.1.29), and

$$d_k = -\mu_k g_k, \qquad g_k = J(x_k)^T f(x_k), \qquad \mu_k = \|g_k\|_2^2 / \|J(x_k)g_k\|_2^2.$$

The choice of β_k is monitored by a parameter Δ_k, which equals the maximum allowed step size. The algorithm also includes a prescription for updating Δ_k. Powell chooses x_{k+1} as follows:

i. If $\|h_k\|_2 \le \Delta_k$ then $x_{k+1} = x_k + h_k$.

ii. If $\|g_k\|_2 \le \Delta_k \le \|h_k\|_2$, choose $\beta_k \in (0, 1]$ so that $\|x_{k+1} - x_k\|_2 = \Delta_k$.

iii. Otherwise set $x_{k+1} = x_k + \Delta_k g_k / \|g_k\|_2$.

The convergence is monitored by $\|f(x_k)\|_2$. When convergence is slow, Δ_k can be decreased, giving a bias towards steepest descent. When convergence is fast, Δ_k is increased, giving a bias towards the Newton direction.

Global methods for nonlinear systems may introduce other problems not inherent in the basic Newton method. The modification introduced may lead to slower convergence and even lead to convergence to a point where the equations are not satisfied.

11.1.8 Numerical Continuation Methods

When it is hard to solve the system $f(x) = 0$, or to find an initial approximation, continuation, embedding or homotopy methods are useful tools. Their use to solve nonlinear systems of equations goes back at least as far as Lahaye [1934]. Briefly, the idea is to find a simpler system $g(x) = 0$, for which the solution $x = x_0$ can be obtained without difficulty, and define a convex embedding (or **homotopy**)

$$H(x, t) = tf(x) + (1 - t)g(x), \tag{11.1.31}$$

so that

$$H(x, 0) = g(x), \qquad H(x, 1) = f(x).$$

If the functions $f(x)$ and $g(x)$ are sufficiently smooth then a solution curve $x(t)$ exists, which satisfies the conditions $x(0) = x_0$, and $x(1) = x^*$. One now attempts to trace the solution curve $x(t)$ of (11.1.31) by computing $x(t_j)$ for an increasing sequence of values of t, $0 = t_0 < t_1 < \ldots < t_p = 1$ by solving the nonlinear systems

$$H(x, t_{j+1}) = 0, \qquad j = 0 : p - 1, \tag{11.1.32}$$

by some appropriate method. The starting approximations can be obtained from previous results, e.g.,

$$x_0(t_{j+1}) = x(t_j),$$

or, if $j \ge 1$, by linear interpolation

$$x_0(t_{j+1}) = x(t_j) + \frac{t_{j+1} - t_j}{t_j - t_{j-1}}\big(x(t_j) - x(t_{j-1})\big).$$

This technique can be used in connection with any of the methods previously mentioned. For example, Newton's method can be used to solve (11.1.32)

$$x_{k+1} = x_k - \left(\frac{\partial H(x_k, t_{j+1})}{\partial x}\right)^{-1} H(x_k, t_{j+1}).$$

The step size should be adjusted automatically to approximately minimize the total number of iterations. A simpler strategy is to choose the number of increments M and take a constant step $\Delta t = 1/M$. If m is sufficiently large, then the iterative process will generally converge. However, the method may fail when turning points of the curve with respect of parameter t are encountered. In this case the embedding family has to be changed, or some other special measure must be taken. Poor performance can also occur because t is not well suited for parametrization. Often the arclength s of the curve provides a better parametrization

Embedding has important applications to the nonlinear systems encountered when finite-difference or finite-element methods are applied to nonlinear boundary-value problems; see Chapter 14. It is also an important tool in nonlinear optimization, e.g., in interior point methods. Often a better choice than (11.1.31) can be made for the embedding, where the systems for $t_j \neq 1$ also contribute to the insight into the questions which originally lead to the system. In elasticity, a technique called **incremental loading** is used, because $t = 1$ may correspond to an unloaded structure for which the solution is known, while $t = 0$ correspond to the actual loading. The technique is also called the **continuation** method.

If the equation $H(x, t) = 0$ is differentiated we obtain

$$\frac{\partial H}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial H}{\partial t} = 0.$$

This gives the differential equation

$$\frac{dx}{dt} = F(x, t), \qquad F(x, t) = -\Big(\frac{\partial H}{\partial x}\Big)^{-1} \frac{\partial H}{\partial t}.$$

Sometimes it is recommended to use a numerical method to integrate this differential equation with initial value $x(1) = x_1$ to obtain the solution curve $x(s)$, and in particular $x(1) = x^*$. However, to use a general purpose method for solving the differential equation is an unnaturally complicated approach. One should instead numerically integrate (11.1.32) very coarsely and then locally use a Newton-type iterative method for solving (11.1.31) as a corrector. This has the advantage that one takes advantage of the fact that the solution curve consists of solutions of (11.1.32), and uses the resulting strong contractive properties of Newton's method. Such predictor corrector continuation methods have been very successful, see Allgower and Georg [1, 1990]. The following algorithm uses Euler's method for integration as a predictor step and Newton's method as a corrector:

Algorithm 11.1.1 *Euler-Newton Method*

Assume that $g(x_0) = 0$. Let $t_0 = 0$, and $h_0 > 0$ be an initial step length.

$x := x_0; \quad t_1 = t_0 + h_0;$
for $j = 1, 2, \ldots,$
  $x_j := x_{j-1} + h_{j-1} F(x_{j-1}, t_{j-1});$ Euler step
  repeat
    $x_j := x_j - (H'(x_j, t_j))^{-1} H(x_j, t_j);$ Newton step
  until convergence
  if $t_j \equiv 1$ **then** *stop*
  else $t_{j+1} = t_j + h_j; \quad h_j > 0;$ new steplength
end

Note the possibility of using the same Jacobian in several successive steps. The convergence properties of the Euler-Newton Method and other predictor-corrector methods are discussed in Allgower and Georg [1, 1990].

Review Questions

1. Describe the nonlinear Gauss–Seidel method.
2. Describe Newton's method for solving a nonlinear system of equations.
3. In order to get global convergence Newton's method has to be modified. Two different approaches are much used. Describe the main features of these modifications.
4. For large n the main cost of an iteration step in Newton's method is the evaluation and factorizing of the matrix of first derivatives. Describe some ways to reduce this cost.
5. Define what is meant by the completeness of a space, a Banach space, a Lipschitz constant and a contraction. Formulate the Contraction Mapping Theorem. You don't need to work out the full proof, but tell where in the proof the completeness is needed.
6. Give the essential features of the assumptions needed in the theorem in the text which is concerned with the convergence of Newton's method for a nonlinear system. What is the order of convergence for simple roots?
7. Describe the essential features of numerical continuation methods for solving a nonlinear system $f(x) = 0$. How is a suitable convex embedding constructed?

Problems

1. The fixed point iteration in Example 11.1.1 can be written $u_{k+1} = \phi(u_k)$, where $u = (x, y)^T$. Compute $\|\phi(u^*)\|_\infty$ for the two roots $u^* = (1, 0)^T$ and $(0, 1)^T$, and use the result to explain the observed convergence behavior.

2. Consider the system of equations

$$x_1^2 - x_2 + \alpha = 0,$$
$$-x_1 + x_2^2 + \alpha = 0.$$

Show that for $\alpha = 1, 1/4$, and 0 there is no solution, one solution, and two solutions, respectively.

3. (a) Describe graphically in the (x, y)-plane nonlinear Gauss–Seidel applied to the system $f(x, y) = 0$, $g(x, y) = 0$. Consider all four combinations of orderings for the variables and the equations.

(b) Do the same thing for nonlinear Jacobi. Consider both orderings of the equations.

4. The system of equations

$$x = 1 + h^2(e^{y\sqrt{x}} + 3x^2)$$
$$y = 0.5 + h^2 \tan(e^x + y^2),$$

can, for small values of h, be solved by fixed point iteration. Write a program which uses this method to solve the system for $h = 0, 0.01, \ldots, 0.10$. For $h = 0$ take $x_0 = 1$, $y_0 = 0.5$, else use the solution for the previous value of h. The iterations should be broken off when the changes in x and y are less than $0.1h^4$.

5. For each of the roots of the system in Example 11.1.1,

$$x^2 - 2x - y + 1 = 0$$
$$x^2 + y^2 - 1 = 0$$

determine whether or not the following iterations are locally convergent:

(a) $x^{k+1} = (1 - y_k^2)^{1/2}$, $\quad y_{k+1} = (x_k - 1)^2$.

(b) $x_{k+1} = y_k^{1/2} + 1$, $\quad y_{k+1} = (1 - x_k^2)$.

6. Apply two iterations of Newton's method to the equations of Problem 5, using the initial approximations $x_0 = 0.1$, and $y_0 = 1.1$.

7. If some of the equations in the system $f(x) = 0$ are linear, Newton's method will take this into account. Show that if (say) $f_i(x)$ is linear, then the Newton iterates x_k, $k \geq 1$, will satisfy $f_i(x_k) = 0$.

Figure 11.1.1. *A rotating double pendulum.*

8. A double pendulum rotates with angular velocity ω around a vertical axis (like a centrifugal regulator). At equilibrium the two pendulums make the angles x_1 and x_2

to the vertical axis, see Fig. 11.1.1. It can be shown that the angles are determined by the equations

$$\tan x_1 - k(2\sin x_1 + \sin x_2) = 0,$$
$$\tan x_2 - 2k(\sin x_1 + \sin x_2) = 0.$$

where $k = l\omega^2/(2g)$.

(a) Solve by Newton's method the system for $k = 0.3$, with initial guesses $x_1 = 0.18$, $x_2 = 0.25$. How many iterations are needed to obtain four correct decimals?

(b) Determine the solutions with four correct decimals and plot the results for

$$k = 0.30, 0.31, \ldots, 0.35, 0.4, 0.5, \ldots, 0.9, 1, 2, 3, 4, 5, 10, 15, 20, \infty.$$

Use the result obtained for the previous k as initial guess for the new k. Record also how many iterations are needed for each value of k.

(c) Verify that the Jacobian is singular for $x_1 = x_2 = 0$, when $k = 1-1/\sqrt{2} \approx 0.2929$. A somewhat sloppy theory suggests that

$$x_1 \approx x_2 \approx \sqrt{k - (1 - 1/\sqrt{2})}, \qquad 0 \le k - (1 - 1/\sqrt{2}) \ll 1.$$

Do your results support this theory?

9. Describe how to apply the Newton idea to the solution of the steady state of a Matrix Riccati equation, i.e., to the solution of a matrix equation of the form,

$$A + BX + XC + XDX = 0,$$

where A, B, C, D are rectangular matrices of appropriate size. Assume that an algorithm for equations of the form $PX + XQ = R$ is given. Under what condition does such a linear equation have a unique solution? You don't need to discuss how to find the first approximation.

10. (a) Derive the formula for $\min B(\eta)$ and the optimal choice of η for the *uncentered* difference approximation to $g'(x)v$, also the simplified error estimate (for $\xi = \frac{1}{2}\|g\|/\|g'\|$).

(b) Work out the details of the study of the directional second derivative.

11. Investigate, for various functions f, g, the ratio of values of $B(\eta)$, obtained with the optimal η and with the value of η derived from he simplified estimate of ξ. Take, for example, $g(x) = e^{\alpha x}$, $g(x) = x^{-k}$.

12. Suppose that $x \in \mathbf{R}^n$, where n is divisible by 3, and that the Jacobian is a square *tridiagonal* matrix.

(a) Design an algorithm, where all the elements of the Jacobian are found by four evaluations of $g(x)$, when the uncentered difference approximation is used.

(b) You may obtain the elements packed in three vectors. How do you unpack them into an $n \times n$ matrix? How many function evaluations do you need with the centered difference approximation?

(c) Generalize to the case of an arbitrary *banded Jacobian*.

Comment: This idea was first published by Curtis, Powell, and Reid [6]

11.2 Nonlinear Least Squares Problems

11.2.1 Introduction

In this section we discuss the numerical solution of nonlinear least squares problem. Let $f: \mathbf{R}^n \to \mathbf{R}^m$, $m \geq n$, and consider the problem

$$\min_{x \in \mathbf{R}^n} \phi(x), \qquad \phi(x) = \tfrac{1}{2}\|f(x)\|_2^2 = \tfrac{1}{2} f(x)^T f(x). \tag{11.2.1}$$

This is a special case of the general optimization problem in $\mathbf{R}^n$ studied in Section 11.3. We will in the following mainly emphasize those aspects of the problem (11.2.1), which derive from the special form of $\phi(x)$. (Note that the nonlinear system $f(x) = 0$ is equivalent to (11.2.1) with $m = n$.)

Fitting data to a mathematical model is an important source of nonlinear least squares problems. Here one attempts to fit given data (y_i, t_i), $i = 1 : m$ to a model function $y = h(x, t)$. If we let $r_i(x)$ represent the error in the model prediction for the i:th observation,

$$r_i(x) = y_i - h(x, t_i), \qquad i = 1, \ldots, m,$$

we want to minimize some norm of the vector $r(x)$. The choice of the least squares measure is justified here, as for the linear case, by statistical considerations. If the observations have equal weight, this leads to the minimization problem in (11.2.1) with $f(x) = r(x)$.

Example 11.2.1.

Exponential fitting problems occur frequently–e.g., the parameter vector x in the expression

$$y(t, x) = x_1 + x_2 e^{x_4 t} + x_3 e^{x_5 t}$$

is to be determined to give the best fit to m observed points (t_i, y_i), $i = 1 : m$, where $m > 5$. Here $y(t, x)$ is linear the parameters x_4, x_5. but nonlinear in x_4, x_5. Hence this problem cannot be handled by the methods in Chapter 7. Special methods for problems which are nonlinear only in some of the parameters are given in Section 11.2.5.

The standard methods for the nonlinear least squares problem require derivative information about the component functions of $f(x)$. We assume here that $f(x)$ is twice continuously differentiable. It is easily shown that the gradient of $\phi(x) = \frac{1}{2} f^T(x) f(x)$ is

$$g(x) = \nabla \phi(x) = J(x)^T f(x), \tag{11.2.2}$$

where

$$J(x)_{ij} = \frac{\partial f_i(x)}{\partial x_j} \in \mathbf{R}^{m \times n}, \quad i = 1 : m, \quad j = 1 : n.$$

is the Jacobian matrix of $f(x)$. The Hessian matrix is

$$H(x) = \nabla^2 \phi(x) = J(x)^T J(x) + Q(x), \quad Q(x) = \sum_{i=1}^{m} f_i(x) G_i(x), \tag{11.2.3}$$

where $G_i(x) \in \mathbf{R}^{n\times n}$, is the Hessian matrix of $f_i(x)$ with elements

$$G_i(x)_{jk} = \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}, \quad i = 1:m, \quad j,k = 1:n. \tag{11.2.4}$$

The special forms of the gradient $g(x)$ and Hessian $H(x)$ can be exploited by methods for the nonlinear least squares problem.

A necessary condition for x^* to be a local minimum of $\phi(x)$ is that x^* is a stationary point, i.e., satisfies

$$g(x^*) = J(x^*)^T f(x^*) = 0. \tag{11.2.5}$$

A necessary condition for a stationary point x^* to be a local *minimum* of $\phi(x)$ is that the Hessian matrix $H(x)$ is positive definite at x^*.

There are basically two different ways to view problem (11.2.1). One could think of this problem as arising from an overdetermined system of nonlinear equations $f(x) = 0$. It is then natural to approximate $f(x)$ by a linear model around a given point x_k

$$\tilde{f}(x) = f(x_k) + J(x_k)(x - x_k), \tag{11.2.6}$$

and use the solution p_k to the linear least squares problem

$$\min_p \|f(x_k) + J(x_k)p\|_2. \tag{11.2.7}$$

to derive an new (hopefully improved) improved approximate solution $x_{k+1} = x_k + p_k$. This approach, which only uses first order derivative information about $f(x)$, leads to a class of methods called **Gauss–Newton** type methods. These methods, which in general only have linear rate of convergence, will be discussed in Section 11.2.2.

In the second approach (11.2.1) is viewed as a special case of unconstrained optimization in $\mathbf{R^n}$. A quadratic model at a point x_k is used,

$$\tilde{\phi}_c(x) = \phi(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k), \tag{11.2.8}$$

where the gradient and Hessian of $\phi(x) = \frac{1}{2}f^T(x)f(x)$ are given by (11.2.2) and (11.2.3). The minimizer of $\tilde{\phi}_c(x)$ is given by $x_{k+1} = x_k + p_k$, where

$$p_k = -H(x_k)^{-1}J(x_k)^T f(x_k). \tag{11.2.9}$$

This method is equivalent to Newton's method applied to (11.2.1), which usually is locally quadratically convergent.

The Gauss–Newton method can be thought of as arising from neglecting the second derivative term

$$Q(x) = \sum_{i=1}^{m} f_i(x)G_i(x),$$

in the Hessian $H(x_k)$. Note that $Q(x_k)$ will be small close to the solution x^* if either the residual norm $\|f(x^*)\|$ is small or if $f(x)$ is only mildly nonlinear. The

behavior of the Gauss–Newton method can then be expected to be similar to that of Newton's method. In particular for a consistent problem where $f(x^*) = 0$ the local convergence will be the same for both methods. However, for moderate to large residual problems the local convergence rate for the Gauss–Newton method can be much inferior to that of Newton's method.

The cost of computing and storing the mn^2 second derivatives (11.2.4) in $Q(x)$ can be prohibitively high. Note, however, that for curve fitting problems the function values $f_i(x) = y_i - h(x, t_i)$ and the derivatives $\partial^2 f_i(x)/\partial x_j \partial x_k$, can be obtained from the single function $h(x, t)$. If $h(x, t)$ is composed of, e.g., simple exponential and trigonometric functions then the Hessian can sometimes be computed cheaply. Another case when it may be feasible to store approximations to all $G_i(x)$, $i = 1 : m$, is when every function $f_i(x)$ only depends on a small subset of the n variables. Then both the Jacobian $J(x)$ and the Hessian matrices $G_i(x)$ will be *sparse* and special methods, such as those discussed in Section 6.5 may be applied.

11.2.2 Gauss–Newton-Type Methods

The Gauss–Newton method for problem (11.2.1) is based on a sequence of linear approximations of $f(x)$ of the form (11.2.6). If x_k denotes the current approximation then the Gauss–Newton step d_k is a solution to the linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \qquad d_k \in \mathbf{R}^n. \tag{11.2.10}$$

and the new approximation is $x_{k+1} = x_k + d_k$. The solution d_k is unique if $\text{rank}(J(x_k)) = n$. Since $J(x_k)$ may be ill-conditioned or singular, d_k should be computed by a stable method using, e.g., the QR- or SVD-decomposition of $J(x_k)$.

The Gauss–Newton step $d_k = -J(x_k)^\dagger f(x_k)$ has the following important properties:

(i) d_k is invariant under linear transformations of the independent variable x, i.e., if $\tilde{x} = Sx$, S nonsingular, then $\tilde{d}_k = Sd_k$.

(ii) if $J(x_k)^T f(x_k) \neq 0$ then d_k is a descent direction for $\phi(x) = \frac{1}{2} f^T(x) f(x)$,

The first property is easily verified. To prove the second property we note that

$$d_k^T g(x_k) = -f(x_k)^T J^\dagger(x_k)^T J(x_k)^T f(x_k) = -\|P_{J_k} f(x_k)\|_2^2, \tag{11.2.11}$$

where $P_{J_k} = J(x_k)J^\dagger(x_k) = P_{J_k}^2$ is the orthogonal projection onto the range space of $J(x_k)$. Further if $J(x_k)^T f(x_k) \neq 0$ then $f(x_k)$ is not in the nullspace of $J(x_k)^T$ and it follows that $P_{J_k} f(x_k) \neq 0$. This proves (ii).

The Gauss–Newton method can fail at an intermediate point where the Jacobian is rank deficient or illconditioned. Formally we can take d_k to be the minimum norm solution

$$d_k = -J(x_k)^\dagger f(x_k).$$

In practice it is necessary to include some strategy to estimate the numerical rank of $J(x_k)$, cf. Section 7.3.2 and 7.6.2. That the assigned rank can have a decisive influence is illustrated by the following example:

Example 11.2.2. (Gill, Murray and Wright [16, p. 136])
Let $J = J(x_k)$ and $f(x_k)$ be defined by

$$J = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}, \qquad f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix},$$

where $\epsilon \ll 1$ and f_1 and f_2 are of order unity. If J is considered to be of rank two then the search direction $d_k = s_1$, whereas if the assigned rank is one $d_k = s_2$, where

$$s_1 = -\begin{pmatrix} f_1 \\ f_2/\epsilon \end{pmatrix}, \qquad s_2 = -\begin{pmatrix} f_1 \\ 0 \end{pmatrix}.$$

Clearly the two directions s_1 and s_2 are almost orthogonal and s_1 is almost orthogonal to the gradient vector $J^T f$.

Usually it is preferable to *underestimate* the rank except when $\phi(x)$ is actually close to an ill-conditioned quadratic function. One could also switch to a search direction along the negative gradient $-g_k = -J(x_k)^T f(x_k)$, or use a linear combination

$$d_k - \mu_k g_k, \qquad \mu_k = \|g_k\|_2^2 / \|J(x_k) g_k\|_2^2.$$

as in Powell's method.

The Gauss–Newton method as described above has several advantages. It solves linear problems in just one iteration and has fast convergence on small residual and mildly nonlinear problems. However, it may not be locally convergent on problems that are very nonlinear or have large residuals.

To analyze the rate of convergence of Gauss–Newton type methods let $J^\dagger(x)$ denote the pseudoinverse of $J(x)$, and assume that $\operatorname{rank}(J(x)) = n$. Then $I = J^\dagger(x)J(x)$, and (11.2.3) can be written in the form

$$H(x) = J(x)^T (I - \gamma K(x)) J(x), \qquad K(x) = J^\dagger(x)^T G_w(x) J^\dagger(x). \tag{11.2.12}$$

where $\gamma = \|f(x)\|_2 \neq 0$, and

$$G_w(x) = \sum_{i=1}^{m} w_i G_i(x), \qquad w(x) = -\frac{1}{\gamma} f(x). \tag{11.2.13}$$

The matrix $K(x)$ is symmetric, and has a geometric interpretation. It is called the **normal curvature matrix** of the n-dimensional surface $z = f(x)$ in $\mathbf{R}^m$, with respect to the unit normal vector $w(x)$. The quantities $\rho_i = 1/\kappa_i$, where

$$\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_n.$$

are the eigenvalues of $K(x)$, are called the **principal radii of curvature** of the surface.

The Hessian matrix $H(x^*)$ is positive definite and x^* a local minimum if and only if $u^T H(x^*) u > 0$, for all $u \in \mathbf{R}^n \neq 0$. If $\operatorname{rank}(J(x^*)) = n$, it follows that $u \neq 0 \Rightarrow J(x^*)u \neq 0$, and hence $H(x^*)$ is positive definite when $I - \gamma K(x^*)$ is positive definite, i.e., when

$$1 - \gamma \kappa_1 > 0. \tag{11.2.14}$$

If $1 - \gamma\kappa_1 \leq 0$ then the least squares problem has a saddle point at x^* or if also $1 - \gamma\kappa_n < 0$ even a local maximum at x^*.

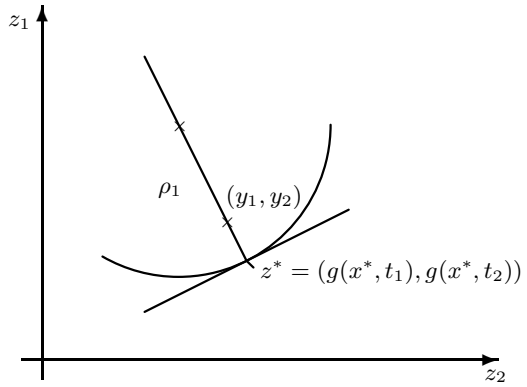

Figure 11.2.1. *Geometry of the data fitting problem for $m = 2$, $n = 1$.*

Example 11.2.3.

The geometrical interpretation of the nonlinear least squares problem (11.2.1) is to find a point on the surface $\{f(x) \mid x \in \mathbf{R}^n\}$ in $\mathbf{R}^m$ closest to the origin. In case of data fitting $f_i(x) = y_i - h(x, t_i)$, and it is more illustrative to consider the surface

$$z(x) = (h(x, t_1), \ldots, h(x, t_m))^T \in \mathbf{R}^m.$$

The problem is then to find the point $z(x^*)$ on this surface closest to the observation vector $y \in \mathbf{R}^m$. This is illustrated in Fig. 11.4.1 for the simple case of $m = 2$ observations and a scalar parameter x. Since in the figure we have $\gamma = \|y - z(x^*)\|_2 < \rho$, it follows that $1 - \gamma\kappa_1 > 0$, which is consistent with the fact that x^* is a local minimum. In general the solution (if it exists) is given by an orthogonal projection of y onto the surface $z(x)$. Compare the geometrical intepretation in Fig. 7.2.1 for the linear case $z(x) = Ax$!

It can be shown that the asymptotic rate of convergence of the Gauss–Newton method in the neighborhood of a critical point x^* is equal to

$$\rho = \gamma \max(\kappa_1, -\kappa_n),$$

where κ_i are the eigenvalues of the of the normal curvature matrix $K(x)$ in (11.2.12) evaluated at x^* and $\gamma = \|f(x^*)\|_2 = 0$. In general convergence is linear, but if $\gamma = 0$ then convergence becomes superlinear. Hence the asymptotic rate of convergence of the undamped Gauss–Newton method is fast when either

(i) the residual norm $\gamma = \|r(x^*)\|_2$ is small, or

(ii) $f(x)$ is mildly nonlinear, i.e. $|\kappa_i|$, $i = 1 : n$ are small.

If x^* is a saddle point then $\gamma\kappa_1 \geq 1$, i.e., using undamped Gauss–Newton one is repelled from a saddle point. This is an excellent property since saddle points are not at all uncommon for nonlinear least squares problems.

The Gauss–Newton method can be modified for global convergence in a similar way as described in Section 11.1.7 Newton's method. If the Gauss–Newton direction d_k is used as a search direction we consider the one-dimensional minimization problem

$$\min_{\lambda} \|f(x_k + \lambda d_k)\|_2^2.$$

As remarked above it is in general not worthwhile to solve this minimization accurately. Instead we can take λ_k to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \ldots$ for which

$$\|f(x_k)\|_2^2 - \|f(x_k + \lambda_k d_k)\|_2^2 \geq \frac{1}{2}\lambda_k \|P_{J_k} f(x_k)\|_2^2.$$

Here $\lambda = 1$ corresponds to the full Gauss–Newton step. Since d_k is a descent direction, this damped Gauss–Newton method is locally convergent on almost all nonlinear least squares problems. In fact is is usually even globally convergent. For large residual or very nonlinear problems convergence may still be slow.

The rate of convergence for the Gauss–Newton method with *exact* line search can be shown to be

$$\tilde{\rho} = \gamma(\kappa_1 - \kappa_n)/(2 - \gamma(\kappa_1 + \kappa_n)).$$

We have $\tilde{\rho} = \rho$ if $\kappa_n = -\kappa_1$ and $\tilde{\rho} < \rho$ otherwise. Since $\gamma\kappa_1 < 1$ implies $\tilde{\rho} < 1$, we always get convergence close to a local minimum. This is in contrast to the undamped Gauss–Newton method, which may fail to converge to a local minimum.

The rate of convergence for the undamped Gauss–Newton method can be estimated during the iterations from

$$\rho_{\text{est}} = \|P_J(x_{k+1})r_{k+1}\|_2 / \|P_J(x_k)r_k\|_2 = \rho + O(\|x_k - x^*\|_2^2). \tag{11.2.15}$$

Since $P_J(x_k)r_k = J(x_k)J(x_k)^\dagger r_k = -J(x_k)p_k$ the cost of computing this estimate is only one matrix-vector multiplication. When $\rho_{\text{est}} > 0.5$ (say) then one should consider switching to a method using second derivative information, or perhaps evaluate the quality of the underlying model.

11.2.3 Trust Region Methods

Even the damped Gauss–Newton method can have difficulties to get around an intermediate point where the Jacobian matrix rank deficient. This can be avoided either by taking second derivatives into account (see Section 11.2.4) or by further stabilizing the damped Gauss–Newton method to overcome this possibility of failure. Methods using the latter approach were first suggested by Levenberg [26, 1944] and Marquardt [24, 1963]. Here a search direction d_k is computed by solving the problem

$$\min_{d_k} \{\|f(x_k) + J(x_k)d_k\|_2^2 + \mu_k \|d_k\|_2^2\}, \tag{11.2.16}$$

where the parameter $\mu_k \geq 0$ controls the iterations and limits the size of d_k. Note that if $\mu_k > 0$ then d_k is well defined even when $J(x_k)$ is rank deficient. As

$\mu_k \to \infty$, $\|d_k\|_2 \to 0$ and d_k becomes parallel to the steepest descent direction. It can be shown that d_k is the solution to the least squares problem with quadratic constraint

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to} \quad \|d_k\|_2 \le \delta_k, \tag{11.2.17}$$

where $\mu_k = 0$ if the constraint in (11.2.17) is not binding and $\mu_k > 0$ otherwise. The set of feasible vectors d_k, $\|d_k\|_2 \le \delta_k$ can be thought of as a region of trust for the linear model $f(x) \approx f(x_k) + J(x_k)(x - x_k)$.

The following trust region strategy has proved very successful in practice:
Let x_0, D_0 and δ_0 be given and choose $\beta \in (0, 1)$. For $k = 0, 1, 2, \ldots$ do

(a) Compute $f(x_k)$, $J(x_k)$, and determine d_k as a solution to the subproblem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to} \quad \|D_k d_k\|_2 \le \delta_k,$$

where D_k is a diagonal scaling matrix.

(b) Compute the ratio $\rho_k = \big(\|f(x_k)\|_2^2 - \|f(x_k + d_k)\|_2^2\big)/\psi_k(d_k)$, where

$$\psi_k(d_k) = \|f(x_k)\|_2^2 - \|f(x_k) + J(x_k)d_k\|_2^2$$

is the model prediction of the decrease in $\|f(x_k)\|_2^2$.

(c) If $\rho_k > \beta$ the step is successful and we set $x_{k+1} = x_k + d_k$, and $\delta_{k+1} = \delta_k$; otherwise set $x_{k+1} = x_k$ and $\delta_{k+1} = \beta\delta_k$. Update the scaling matrix D_k.

The ratio ρ_k measures the agreement between the linear model and the nonlinear function. After an unsuccessful iteration δ_k is reduced. The scaling D_k can be chosen such that the algorithm is scale invariant, i.e., the algorithm generates the same iterations if applied to $r(Dx)$ for any nonsingular diagonal matrix D. It can be proved that if $f(x)$ is continuously differentiable, $f'(x)$ uniformly continuous and $J(x_k)$ bounded then this algorithm will converge to a stationary point.

A trust region implementation of the Levenberg-Marquard method will give a Gauss–Newton step close to the solution of a regular problem. Its convergence will therefore often be slow for large residual or very nonlinear problems. Methods using second derivative information , see Section 11.2.4 are somewhat more efficient but also more complex than the Levenberg-Marquardt methods.

11.2.4 Newton-Type Methods

The analysis in the Section 11.2.2 showed that for large residual problems and strongly nonlinear problems, methods of Gauss–Newton type may converge slowly. Also, these methods can have problems at points where the Jacobian is rank deficient. When second derivatives of $f(x)$ are available Newton's method, which uses the quadratic model (11.2.8), can be used to overcome these problems. The optimal point d_k of this quadratic model, satisfies the linear system

$$H(x_k)d_k = -J(x_k)^T f(x_k), \tag{11.2.18}$$

where $H(x_k)$ is the Hessian matrix at x_k, and $x_k + d_k$ is chosen as the next approximation.

It can be shown, see Dennis and Schnabel [10, 1983, p. 229], that Newton's method is quadratically convergent to a local minimum x^* as long as $H(x)$ is Lipschitz continuous around x_k and $H(x^*)$ is positive definite. To get global convergence a line search algorithm is used, where the search direction d_k is taken as the Newton direction. Note that the Hessian matrix $H(x_k)$ must be positive definite in order for the Newton direction d_k to be a descent direction.

Newton's method is not often used since the second derivative term $Q(x_k)$ in the Hessian is rarely available at a reasonable cost. However, a number of methods have been suggested that partly takes the second derivatives into account, either explicitly or implicitly. An implicit way to obtain second derivative information is to use a general quasi-Newton optimization routine, which successively builds up approximations B_k to the Hessian matrices $H(x_k)$. The search directions are computed from

$$B_k d_k = -J(x_k)^T f(x_k),$$

where B_k satisfies the quasi-Newton conditions

$$B_k s_k = y_k, \qquad s_k = x_k - x_{k-1}, \qquad y_k = g(x_k) - g(x_{k-1}), \tag{11.2.19}$$

where $g(x_k) = J(x_k)^T f(x_k)$. As starting value $B_0 = J(x_0)^T J(x_0)$ is recommended.

The direct application of quasi-Newton methods to the nonlinear least squares problem outlined above has not been so efficient in practice. One reason is that these methods disregard the information in $J(x_k)$, and often $J(x_k)^T J(x_k)$ is the dominant part of $H(x_k)$. A more successful approach is to approximate $H(x_k)$ by $J(x_k)^T J(x_k) + S_k$, where S_k is a quasi-Newton approximation of the term $Q(x_k)$. Initially one takes $S_0 = 0$. The quasi-Newton relations (11.2.19) can now be written

$$S_k s_k = z_k, \qquad z_k = \big(J(x_k) - J(x_{k-1})\big)^T f(x_k), \tag{11.2.20}$$

where S_k is required to be symmetric. It can be shown that a solution to (11.2.20) which minimizes the change from S_{k-1} in a certain weighted Frobenius norm is given by the update formula

$$B_k = B_{k-1} + \frac{w_k y_k^T + y_k w_k^T}{y_k^T s_k} - \frac{w_k^T s_k y_k y_k^T}{y_k^T s_k^2}, \tag{11.2.21}$$

where $s_k = x_k - x_{k-1}$, and $w_k = z_k - B_{k-1} s_k$.

In some cases the updating (11.2.21) gives inadequate results. This motivates the inclusion of "sizing" in which the matrix B_k is replaced by $\tau_k B_k$, where

$$\tau_k = \min\{|s_k^T z_k| / |s_k^T B_k s_k|, 1\}.$$

This heuristic choice ensures that S_k converges to zero for zero residual problems, which improves the convergence behavior.

In another approach, due to Gill and Murray [15], $J(x_k)^T J(x_k)$ is regarded as a good estimate of the Hessian in the right invariant subspace corresponding

to the large singular values of $J(x_k)$. In the complementary subspace the second derivative term $Q(x_k)$ is taken into account. Let the singular value decomposition of $J(x_k)$ be

$$J(x_k) = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T, \qquad \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n),$$

where the singular values are ordered so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. Then putting $Q_k = Q(x_k)$ the equations for the Newton direction $d_k = Vq$ can be written

$$(\Sigma^2 + V^T Q_k V)q = -\Sigma r_1, \qquad r_1 = (\, I_n \quad 0\,)\, U^T f(x_k). \tag{11.2.22}$$

We now split the singular values into two groups, $\Sigma = \mathrm{diag}(\Sigma_1, \Sigma_2)$, where $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ are the "large" singular values. If we partition V, q and $\bar{r}$ conformally, then the first r equations in (11.2.22) can be written.

$$(\Sigma_1^2 + V_1^T Q_k V_2)q_1 + V_1^T Q_k V_2 q_2 = -\Sigma_1 \bar{r}_1.$$

If the terms involving Q_k are neglected compared to $\Sigma_1^2 q_1$ we get $q_1 = -\Sigma_1^{-1}\bar{r}_1$. If this is substituted into the last $(n-r)$ equations we can solve for q_2 from

$$(\Sigma_2^2 + V_2^T Q_k V_2)q_2 = -\Sigma_2 \bar{r}_2 - V_2^T Q_k V_1 q_1.$$

The approximate Newton direction is then given by $d_k = Vq = V_1 q_1 + V_2 q_2$. The splitting of the singular values is updated at each iteration, the idea being to maintain r close to n as long as adequate progress is made.

There are several alternative ways to implement the method by Gill and Murray [15, 1978]. If Q_k is not available explicitly, then a finite difference approximation to $V_2^T Q_k V_2$ can be obtained as follows. Let v_j be a column of V_2 and h a small positive scalar. Then

$$(\nabla r_i(x_k + hv_j) - \nabla r_i(x_k))/h = v_j^T G_i(x_k) + O(h).$$

The vector on the left hand side is the ith row of $(J(x_k+hv_j)-J(x_k))/h$. Multplying with $r_i(x_k)$ and adding we obtain

$$\begin{aligned} r(x_k)^T (J(x_k + hv_j) - J(x_k))/h &= v_j^T \sum_{i=1}^{m} r_i(x_k) G_i(x_k) + O(h) \\ &= v_j^T Q_k + O(h). \end{aligned}$$

Repeating this for all columns in V_2 we obtain an approximation for $V_2^T Q_k$ and we finally form $(V_2^T Q_k)V_2$.

11.2.5 Separable Problems

A nonlinear least squares problem is said to be **separable** if the parameter vector x can be partitioned as $x^T = (y^T, z^T)$, with the subproblem

$$\min_y \|r(y,z)\|_2, \quad y \in \mathbf{R}^p, \quad z \in \mathbf{R}^q, \tag{11.2.23}$$

easy to solve. In the following we restrict ourself to the particular case when $r(y,z)$ is linear in y i.e.

$$r(y,z) = f(z)y - g(z), \quad f(z) \in \mathbf{R}^{m\times p}. \tag{11.2.24}$$

Then the minimum norm solution to (11.2.22) is $y(z) = f^\dagger(z)g(z)$, where $f^\dagger(z)$ is the pseudoinverse of $f(z)$. The original problem can be written

$$\min_z \|g(z) - f(z)y(z)\|_2 = \min \|(I - P_{f(z)})g(z)\|_2 \tag{11.2.25}$$

where $P_{f(z)} = f(z)f(z)^\dagger$ is the orthogonal projector onto the range of $f(z)$. Algorithms based on (11.2.25) are often called **variable projection algorithms**.

Many practical nonlinear least squares problems are separable in this way. A particularly simple case is when $r(y,z)$ is linear in both y and z so that we also have

$$r(y,z) = H(y)z - h(y), \qquad H(y) \in \mathbf{R}^{m\times q}.$$

Example 11.2.4.

Consider the exponential fitting problem

$$\min_{y,z} \sum_{i=1}^{m} (y_1 e^{z_1 t_i} + y_2 e^{z_2 t_i} - g_i)^2.$$

Here the model is nonlinear only in the parameters z_1 and z_2. Given values of z_1 and z_2 the subproblem (11.2.22) is easily solved.

A problem that arises in many applications, e.g., where we have reactions with different time constant is to approximate empirical data with a sum of exponential functions. This problem is often ill-conditioned because the same data can be well approximated by different exponential sums.

We describe a standard method for solving this problem is **Prony's method**. . Assume that that the function $y = f(x)$ is given in equidistant points with the coordinates (x_i, y_i), $i = 1 : m$, where $x_i = x_1 + ih$. We want to approximate these data with a function

$$q(x) = \sum_{j=1}^{n} a_j e^{\lambda_j x}. \tag{11.2.26}$$

Putting $c_j = a_j e^{\lambda_j x_1}$ and $v_j = e^{h\lambda_j}$, we obtain the linear system of equations $Mc = y$, where

$$M = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_n \\ \vdots & \vdots & \cdots & \vdots \\ v_1^m & v_2^m & \cdots & v_n^m \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{11.2.27}$$

Now assume that the unknown $v_1, \ldots, v_n$ are roots to the polynomial

$$\phi(v) = (v - v_1)(v - v_2) \cdots (v - v_n) = v^n + s_1 v^{m-1} + \cdots s_m.$$

Multiplying the equations in (11.2.27) in turn by $s_n, s_{n-1}, \ldots, s_1, s_0 = 1$, and adding, we obtain

$$\sum_{j=1}^{n} \phi(v_j) c_j = \sum_{j=0}^{n} s_{n-j} y_j = 0$$

since $\phi(v_j) = 0$, $j = 1 : n$. Normally n is substantially smaller than m. By shifting the origin with h we get a new equation. Repeating this we get a (usually overdetermined) system Thus we have $m - n + 1$ equations for determining the unknowns $s_n, s_{n-1}, \ldots, s_1$. This can be solved by the method of least squares. Determining the roots of the polynomial $\phi(v)$ we obtain v_j and $\lambda_j = \ln v_j / h$. Finally we get c_j from the linear system (11.2.27) and $a_j = c_j e^{-\lambda_j x_1}$.

We here describe a variable projection algorithm due to Kaufman [21, 1975], which uses a Gauss–Newton method applied to the problem (11.2.25). The algorithm contains two steps merged into one. Let $x_k = (y_k, z_k)^T$ be the current approximation. The next approximation is determined as follows:

(i) Compute the solution δy_k to the linear subproblem

$$\min_{\delta y_k} \left\| f(z_k)\delta y_k - \big(g(z_k) - f(z_k) y_k\big) \right\|_2, \tag{11.2.28}$$

and put $y_{k+1/2} = y_k + \delta_k$, and $x_{k+1/2} = (y_{k+1/2}, z_k)^T$.

(ii) Compute d_k as the Gauss–Newton step at $x_{k+1/2}$, i.e., d_k is the solution to

$$\min_{d_k} \left\| C(x_{k+1/2}) d_k + r(y_{k+1/2}, z_k) \right\|_2, \tag{11.2.29}$$

where the Jacobian is $C(x_{k+1/2}) = \big(f(z_k), r_z(y_{k+1/2}, z_k)\big)$. Take $x_{k+1} = x_k + \lambda_k d_k$ and go to (i).

In (11.2.29) we have used that by (11.2.23) the first derivative of r with respect to y is given by $r_y(y_{k+1/2}, z_k) = f(z_k)$. The derivatives with respect to z are given by

$$r_z(y_{k+1/2}, z_k) = B(z_k) y_{k+1/2} - g'(z_k), \qquad B(z)y = \left(\frac{\partial F}{\partial z_1} y, \ldots, \frac{\partial F}{\partial z_q} y \right),$$

where $B(z)y \in \mathbf{R}^{m \times q}$. Note that in case $r(y, z)$ is linear also in y it follows from (11.2.4) that $C(x_{k+1/2}) = (f(z_k), H(y_{k+1/2}))$. To be robust the algorithms for separable problems must employ a line search or trust region approach for the Gauss–Newton steps as described in Section 11.2.3 and 11.2.4.

It can be shown that the Gauss–Newton algorithm applied to (11.2.25) has the same asymptotic convergence rate as the ordinary Gauss–Newton's method. In

particular both converge quadratically for zero residual problem. This is in contrast to the naive algorithm for separable problems of alternatively minimizing $\|r(y,z)\|_2$ over y and z, which *always* converges linearly. One advantage of the Kaufman algorithm is that *no starting values for the linear parameters have to be provided*. We can, e.g., take $y_0 = 0$ and determine $y_1 = \delta y_1$, in the first step of (11.2.28). This seems to make a difference in the first steps of the iterations, and sometimes the variable projection algorithm can solve problems for which methods not using separability fail.

11.2.6 Orthogonal Distance Regression

Consider the problem of fitting observations (y_i, t_i), $i = 1 : m$ to a mathematical model

$$y = f(p, t). \tag{11.2.30}$$

where y and t are scalar variables and $p \in \mathbf{R}^n$ are parameters to be determined. In the classical regression model the values t_i of the independent variable are assumed to be exact and only y_i are subject to random errors. Then it is natural to minimize the sum of squares of the deviations $y_i - g(p, t_i)$. In this section we consider the more general situation, when also the values t_i contain errors.

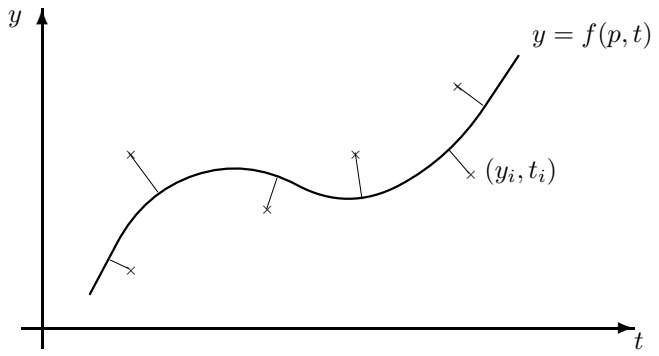

Figure 11.2.2. *Orthogonal distance regression.*

Assume that y_i and t_i are subject to errors $\bar{\epsilon}_i$ and $\bar{\delta}_i$ respectively, so that

$$y_i + \bar{\epsilon}_i = f(p, t_i + \bar{\delta}_i), \qquad i = 1 : m,$$

where $\bar{\epsilon}_i$ and $\bar{\delta}_i$ are independent random variables with zero mean and variance σ^2. Then the parameters p should be chosen so that the sum of squares of the **orthogonal distances** from the observations (y_i, t_i) to the curve in (11.2.30) is minimized, cf. Fig. 11.4.2. Hence the parameters p should be chosen as the solution

to

$$\min_{p,\epsilon,\delta} \sum_{i=1}^{m} (\epsilon_i^2 + \delta_i^2), \quad \text{subject to} \quad y_i + \epsilon_i = f(p, t_i + \delta_i), \quad i = 1:m.$$

Eliminating ϵ_i using the constraints we arrive at the **orthogonal distance problem**

$$\min_{p,\delta} \sum_{i=1}^{m} \big(f(p, t_i + \delta_i) - y_i\big)^2 + \delta_i^2. \tag{11.2.31}$$

Note that (11.2.31) is a nonlinear least squares problem even if $f(p,t)$ is linear in p.

The problem (11.2.31) has $(m+n)$ unknowns p and δ. In applications usually $m \gg n$ and accounting for the errors in t_i will considerably increase the size of the problem. Therefore the use of standard methods will not be efficient unless the special structure is taken into account to reduce the work. If we define the residual vector $r(\delta, p) = (r_1^T(\delta, p), r_2^T(\delta))$ by

$$r_1^T(\delta, p)_i = f(p, t_i + \delta_i) - y_i, \qquad r_2^T(\delta) = \delta_i, \quad i = 1:m,$$

the Jacobian matrix for problem (11.2.31) can be written in block form as

$$\tilde{J} = \begin{pmatrix} D_1 & J \\ \underbrace{I_m}_{m} & \underbrace{0}_{n} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix} \in \mathbf{R}^{2m \times (m+n)}, \tag{11.2.32}$$

where

$$D_1 = \operatorname{diag}(d_1, \ldots, d_m), \quad d_i = \left(\frac{\partial f}{\partial t}\right)_{t=t_i+\delta_i},$$
$$J_{ij} = \left(\frac{\partial f}{\partial p_j}\right)_{t=t_i+\delta_i}, \quad i = 1:m, \quad j = 1:n.$$

Note that $\tilde{J}$ is sparse and highly structured. In the Gauss–Newton method we compute corrections $\Delta\delta_k$ and Δp_k to the current approximations which solve the linear least squares problem

$$\min_{\Delta\delta, \Delta p} \left\| \tilde{J} \begin{pmatrix} \Delta\delta \\ \Delta p \end{pmatrix} - \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \right\|_2, \tag{11.2.33}$$

where $\tilde{J}, r_1$, and r_2 are evaluated at the current estimates of δ and p. To solve this problem we need the QR decomposition of $\tilde{J}$. This can be computed in two steps. First we apply a sequence of Givens rotations $Q_1 = G_m \cdots G_2 G_1$, where $G_i = R_{i,i+m}$, $i = 1:m$, to zero the (2,1) block of $\tilde{J}$:

$$Q_1 \tilde{J} = \begin{pmatrix} D_2 & K \\ 0 & L \end{pmatrix}, \qquad Q_2 \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix},$$

where D_2 is again a diagonal matrix. The problem (11.2.33) now decouples, and Δp_k is determined as the solution to

$$\min_{\Delta p} \|L\Delta p - s_2\|_2.$$

Here $L \in \mathbf{R}^{m\times n}$, so this is a problem of the same size as that which defines the Gauss–Newton correction in the classical nonlinear least squares problem. We then have

$$\Delta\delta_k = D_2^{-1}(s_2 - K\Delta p_k).$$

So far we have assumed that y and t are scalar variables. More generally if $y \in \mathbf{R}^{n_y}$ and $t \in \mathbf{R}^{n_t}$ the problem becomes

$$\min_{p,\delta} \sum_{i=1}^{m} \Big(\|f(p, t_i + \delta_i) - y_i\|_2^2 + \|\delta_i\|_2^2 \Big).$$

The structure in this more general problem can also be taken advantage of in a similar manner.

Schwetlik and Tiller [33, 1985] use a partial Marquardt type regularization where only the Δx part of $\tilde{J}$ is regularized. The algorithm by Boggs, Byrd and Schnabel [1985] incorporates a full trust region strategy. Algorithms for the nonlinear case, based on stabilized Gauss–Newton methods, have been given by Schwetlik and Tiller [1986] and Boggs, Byrd and Schnabel [1986].

11.2.7 Fitting of Circles and Ellipses.

A special nonlinear least squares problem that arises in many areas of applications is to fit given data points to a geometrical element, which may be defined in implicit form. We have already discussed fitting data to an affine linear manifold such as a line or a plane. The problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics.

Least squares algorithms to fit an by $f(x, y, p)$ implicitly defined curve in the x-y plane can be divided into two classes. In the first, called **algebraic fitting**, a least squares functional is used, which directly involves the function $f(x, y, p) = 0$ to be fitted, If (x_i, y_i), $i = 1 : n$ are given data points we minimize the functional

$$\Phi(p) = \sum_{i=1}^{m} f^2(x_i, y_i, p).$$

The second method, geometric fitting, minimizes a least squares functional involving the geometric distances from the data points to the curve; cf. orthogonal distance regression. Often algebraic fitting leads to a simpler problem, in particular when f is linear in the parameters p.

We first discuss algebraic fitting of circles. A circle has three degrees of freedom and can be represented algebraically by

$$f(x, y, p) = a\ (\,x\ y\,) \begin{pmatrix} x \\ y \end{pmatrix} + (b_1\ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0.$$

We define a parameter vector p and an $m \times 4$ matrix S with rows s_i^T by

$$p = (a, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2 + y_i^2, x_i, y_i, 1). \tag{11.2.34}$$

The problem can now be formulated as

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1.$$

Note that the p is defined only up to a constant multiple, which is why the constraint is required. The solution equals the right singular vector corresponding to the smallest singular value of S. When p is known the center z and radius ρ of the circle can be obtained from

$$z = -\frac{1}{2a}\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \qquad \rho = \frac{1}{2a}\sqrt{\|b\|_2^2 - 4ac}. \tag{11.2.35}$$

We now discuss the algebraic fitting of ellipses. An ellipse in the x-y plane can be represented algebraically by

$$f(x, y, p) = (x\ y)\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + (b_1\ b_2)\begin{pmatrix} x \\ y \end{pmatrix} + c = 0. \tag{11.2.36}$$

It we define

$$p = (a_{11}, a_{12}, a_{22}, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2, 2x_iy_i, y_i^2, x_i, y_i, 1), \tag{11.2.37}$$

then we have $\Phi(p) = \|Sp\|_2^2$, where S is an $m \times 6$ matrix with rows s_i^T. Obviously the parameter vector is only determined up to a constant factor. Hence, we must complete the problem formulation by including some constraint on p. Three such constraints have been considered for fitting ellipses.

(a) **SVD constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1. \tag{11.2.38}$$

The solution of this constrained problem equals the right singular vector corresponding to the smallest singular value of S.

(b) **Linear constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad p^T b = 1, \tag{11.2.39}$$

where b is a fixed vector. Assuming $\|b\|_2 = 1$, which is no restriction, and let H be an orthogonal matrix such that $Hb = e_1$. Then the constraint becomes $(Hp)^T e_1 = 1$ so we can write $Sp = (SH^T)(Hp)$, where $Hp = (1q^T)^T$. Now if we partition $SH^T = [sS_2]$ we arrive at the unconstrained problem

$$\min_q \|S_2 q + s\|_2^2, \tag{11.2.40}$$

which is a standard linear least squares problem.

(c) **Quadratic constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|Bp\|_2 = 1. \tag{11.2.41}$$

Of particular interest is the choice $B = (0\ I)$. In this case, if we let $p^T = (p_1, p_2)$ the constraint can be written $\|p_2\|_2^2 = 1$, and is equivalent to a generalized total least squares problem. The solution can then be obtained as follows. First form the QR decomposition of S,

$$S = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

We can now determine p_2 from the SVD of S and then p_1 from back-substitution in $R_{11}p_1 = -R_{12}p_2$.

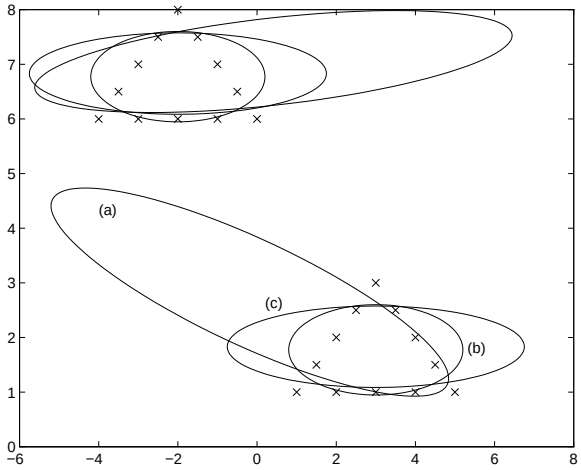

Figure 11.2.3. *Ellipse fits for triangle and shifted triangle data: (a) SVD constraint; (b) Linear constraint $\lambda_1 + \lambda_2 = 1$; (c) Bookstein constraint $\lambda_1^2 + \lambda_2^2 = 1$.*

It should be stressed that the different constraints above can lead to very different solutions, unless the errors in the fit are small. One desirable property of the fitting algorithm is that when the data is translated and rotated the fitted ellipse should be transformed in the same way. It can be seen that to lead to this kind of invariance the constraint must involve only symmetric functions of the eigenvalues of the matrix A.

The disadvantage of the SVD constraint is its non-invariance under translation and rotations. In case of a linear constraint the choice $b^T = (1\ 0\ 1\ 0\ 0\ 0)$, which corresponds to

$$\text{trace}(A) = a_{11} + a_{22} = \lambda_1 + \lambda_2 = 1. \tag{11.2.42}$$

gives the desired invariance. This constraint, attributed to Bookstein,

$$\|A\|_F^2 = a_{11}^2 + 2a_{12}^2 + a_{22}^2 = \lambda_1^2 + \lambda_2^2 = 1. \tag{11.2.43}$$

also leads to this kind of invariance. Note that the Bookstein constraint can be put in the form $(0\ I)$ by permuting the variables and scaling by $\sqrt{2}$.

To construct and plot the ellipse it is convenient to convert the algebraic form (11.2.36) to the parametric form

$$\begin{pmatrix} x(\theta) \\ y(\theta) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + Q(\alpha) \begin{pmatrix} a\,\cos(\theta) \\ b\,\sin(\theta) \end{pmatrix}, \quad Q(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \quad (11.2.44)$$

The new parameters (x_c, y_c, a, b, α) can be obtained from the algebraic parameters p. The eigendecomposition $A = Q\Lambda Q^T$, where A is the 2×2 matrix in (11.2.36) can be obtained by a Jacobi rotation, see Section 10.4.1. We assume that $a_{12} = 0$ since otherwise $Q = I$ and $\Lambda = A$ is the solution. To determine Λ and Q we first compute

$$\tau = (a_{22} - a_{11})/(2a_{12}), \qquad \tan\alpha = t = \text{sign}\,(\tau)/\big(|\tau| + \sqrt{1+\tau^2}\big).$$

The elements in Q and Λ are then given by

$$\begin{aligned} \cos\alpha = 1/\sqrt{1+t^2}, \qquad & \sin\alpha = t\,\cos\alpha, \\ \lambda_1 = a_{11} - t\,a_{12}, \qquad & \lambda_2 = a_{22} + t\,a_{12}. \end{aligned}$$

If we introduce the new coordinates $z = Q\tilde{z} + s$ in the algebraic form (11.2.36) this equation becomes

$$\tilde{z}^T\Lambda\tilde{z} + (2As + b)^T Q\tilde{z} + (As + b)^T s + c = 0.$$

Here s can be chosen so that this equation reduces to

$$\lambda_1\tilde{x}^2 + \lambda_2\tilde{y}^2 + \tilde{c} = 0.$$

Hence the center s equals

$$s = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = -\frac{1}{2}A^{-1}b = -\frac{1}{2}A^{-1}Q\Lambda^{-1}(Q^T b), \quad (11.2.45)$$

and the axis (a, b) of the ellipse are given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sqrt{-\tilde{c}}\,\text{diag}\,\Lambda^{-1/2}, \qquad \tilde{c} = c + \frac{1}{2}b^T s = -\frac{1}{2}\tilde{b}^T\Lambda^{-1}\tilde{b}. \quad (11.2.46)$$

In geometric fitting of data (x_i, y_i), $i = 1 : m$ to a curve of the form $f(x, y, p) = 0$, where the orthogonal distance $d_i(p)$ is first measured from each data point to the curve, where

$$d_i^2(p) = \min_{f(x,y,p)=0} \big((x - x_i)^2 + (y - y_i)^2\big).$$

Then the problem

$$\min_p \sum_{i=1}^{m} d_i^2(p)$$

is solved. This is similar to orthogonal distance regression described for an explicitly defined function $y = f(x, \beta)$ in Section 11.2.6. Algorithms for **geometric fitting** are described in Gander, Golub, and Strebel [13, 1994].

For implicitly defined functions the calculation of the distance function $d_i(p)$ is more complicated than for explicit functions. When the curve admits a parametrization as in the case of the ellipse the minimization problem for each point is only one-dimensional.

We consider first the orthogonal distance fitting of a circle written in parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c - r\cos\phi \\ y - y_c - r\sin\phi \end{pmatrix} = 0, \tag{11.2.47}$$

where $p = (x_c, y_c, r)^T$. The problem can be written as a nonlinear least squares problem

$$\min_{p,\phi_i} \|r(p, \phi)\|_2^2, \quad \phi = (\phi_1, \ldots, \phi_m), \tag{11.2.48}$$

where

$$r = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbf{R}^{2m}, \quad r_i = \begin{pmatrix} x_i - x_c - r\cos\phi_i \\ y_i - y_c - r\sin\phi_i \end{pmatrix}.$$

We have $2m$ nonlinear equations for $m+3$ unknowns $\phi_1, \ldots, \phi_m$ and x_c, y_c, r. (Note that at least 3 points are needed to define a circle.)

We now show how to contruct the Jacobian matrix, which should be evaluated at the current approximations to the $m + 3$. parameters. We need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = r \begin{pmatrix} \sin\phi_i \\ -\cos\phi_i \end{pmatrix}, \qquad \frac{\partial r_i}{\partial r} = -\begin{pmatrix} \cos\phi_i \\ \sin\phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial x_c} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \qquad \frac{\partial r_i}{\partial y_c} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

After reordering the rows the Jacobian associated with this problem has the form

$$J = \begin{pmatrix} rS & A \\ \underbrace{-rC}_{m} & \underbrace{B}_{3} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix},$$

where

$$S = \operatorname{diag}(\sin\phi_i), \qquad C = \operatorname{diag}(\cos\phi_i), \tag{11.2.49}$$

are two $m \times m$ diagonal matrices. Here the first block column, which corresponds to the m parameters ϕ_i, is orthogonal. Multiplying from the left with an orthogonal matrix we obtain

$$Q^T J = \begin{pmatrix} rI & SA - CB \\ 0 & CA + SB \end{pmatrix}, \qquad Q = \begin{pmatrix} S & C \\ -C & S \end{pmatrix}.$$

To obtain the QR factorization of J we only need to compute the QR factorization of the $m \times 3$ matrix $CA + SB$.

A Gauss–Newton type method with a trust region strategy can be implemented using this QR decomposition of the Jacobian. Good starting values for the parameters may often be obtained using an algebraic fit as described in the previous

section. Experience shows that the amount of computation involved in a geometric fit is at least an order of magnitude more than for an algebraic fit.

For the geometric fit of an ellipse we use the parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a\,\cos\phi \\ b\,\sin\phi \end{pmatrix} = 0. \tag{11.2.50}$$

where $p = (x_c, y_c, a, b, \alpha)^T$ and

$$Q(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}.$$

The problem can be written as a nonlinear least squares of the form (11.2.48), where

$$r_i = \begin{pmatrix} x_i - x_c \\ y_i - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a\,\cos\phi_i \\ b\,\sin\phi_i \end{pmatrix}.$$

We thus have $2m$ nonlinear equations for $m+5$ unknowns $\phi_1, \ldots, \phi_m$ and x_c, y_c, a, b, α. To construct the Jacobian we need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = Q(\alpha) \begin{pmatrix} -a\,\sin\phi_i \\ b\,\cos\phi_i \end{pmatrix}, \qquad \frac{\partial r_i}{\partial \alpha} = -\frac{d}{d\alpha} Q(\alpha) \begin{pmatrix} a\,\cos\phi_i \\ b\,\sin\phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial a} = -Q(\alpha) \begin{pmatrix} \cos\phi_i \\ 0 \end{pmatrix}, \qquad \frac{\partial r_i}{\partial b} = -Q(\alpha) \begin{pmatrix} 0 \\ \sin\phi_i \end{pmatrix}.$$

Note that

$$\frac{d}{d\alpha} Q(\alpha) = \begin{pmatrix} -\sin\alpha & \cos\alpha \\ -\cos\alpha & -\sin\alpha \end{pmatrix} = Q \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

After a reordering of the rows the Jacobian associated with this problem has the form

$$J = U \begin{pmatrix} \underbrace{-aS}_{m} & \underbrace{A}_{3} \\ \underbrace{bC}_{m} & \underbrace{B}_{3} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix}, \quad S = \operatorname{diag}(\sin\phi_i), \quad C = \operatorname{diag}(\cos\phi_i).$$

where $U = -\operatorname{diag}(Q, \ldots, Q) \in \mathbf{R}^{2m \times 2m}$ is a block diagonal orthogonal matrix and S and C given by (11.2.49). The ith row of the matrices $A \in \mathbf{R}^{m \times 5}$ and $B \in \mathbf{R}^{m \times 5}$ are

$$\begin{aligned} a_i^T &= (\, -b\sin\phi_i \quad \cos\phi_i \quad 0 \quad \cos\alpha \quad \sin\alpha\,), \\ b_i^T &= (\, a\cos\phi_i \quad 0 \quad \sin\phi_i \quad -\sin\alpha \quad \cos\alpha\,). \end{aligned}$$

The first m columns of $U^T J$ can be diagonalized using a sequence of Givens rotations, where the ith rotation zeros the second component in the vector

$$\begin{pmatrix} -a\,\sin\phi_i \\ b\,\cos\phi_i \end{pmatrix}, \quad i = 1 : m.$$

The fitting of a sphere or an ellipsoid can be treated analogously. The sphere can be represented in parametric form as

$$f(x, y, z, p) = \begin{pmatrix} x - x_c - r\cos\theta\cos\phi \\ y - y_c - r\cos\theta\sin\phi \\ z - z_c - r\sin\theta \end{pmatrix} = 0, \tag{11.2.51}$$

where $p = (x_c, y_c, z_c, r)^T$. We get $3m$ nonlinear equations for $2m + 4$ unknowns. The first $2m$ columns of the Jacobian matrix can simply be brought into upper triangular form; cf. Computer Exercies 2.

When the data covers only a small arc of the circle or a small patch of the sphere the fitting problem can be ill-conditioned. An important application involving this this type of data is the fitting of a spherical lens. Also the fitting of a sphere or an ellipsoid to near planar data gives rise to ill-conditioned problems.

Review Questions

1. Describe the damped Gauss–Newton method with a recommended step length procedure.
2. How does the Gauss–Newton method differ from the full Newton method? When can the behavior of the Gauss–Newton method be expected to be similar to that of Newton's method?
3. What is a separable nonlinear least squares problem? Describe a recommended method. Give an important example.
4. Consider fitting observations (y_i, t_i), $i = 1 : m$ to the model $y = g(p, t)$, where y and t are scalar variables and $p \in \mathbf{R}^n$ are parameters to be determined. Formulate the method of orthogonal distance regression for this problem.

Computer Exercises

1. One wants to fit a circle with radius r and center (x_0, y_0) to given data (x_i, y_i), $i = 1 : m$. The orthogonal distance from (x_i, y_i) to the circle

$$d_i(x_0, y_0, r) = r_i - r, \qquad r_i = \left((x_i - x_0)^2 + (y_i - y_0)^2\right)^{1/2},$$

depends nonlinearly on the parameters x_0, y_0. The problem

$$\min_{x_0, y_0, r} \sum_{i=1}^{m} d_i^2(x_0, y_0, r)$$

is thus a nonlinear least squares problem. An approximative linear model is obtained by writing the equation of the circle $(x - x_0)^2 + (y - y_0)^2 = r^2$ in the form

$$\delta(x_0, y_0, c) = 2xx_0 + 2yy_0 + c = x^2 + y^2,$$

which depends linearly on the parameters x_0, y_0 and $c = r^2 - x_0^2 - y_0^2$. If these parameters are known, then the radius of the circle can be determined by $r =$

$(c + x_0^2 + y_0^2)^{1/2}$.

(a) Write down the overdetermined linear system $\delta_i(x_0, y_0, c) = x^2+y^2$ corresponding to the data $(x, y) = (x_i, y_i)$, where

$$\begin{array}{cccccccc} x_i & 0.7 & 3.3 & 5.6 & 7.5 & 0.3 & -1.1 \\ y_i & 4.0 & 4.7 & 4.0 & 1.3 & -2.5 & 1.3 \end{array}$$

(b) Describe, preferably in the form of a MATLAB program a suitable algorithm to calculate x_0, y_0, c with the linearized model. The program should function for all possible cases, e.g., even when $m < 3$.

2. Generalize the algorithm described in Sec. 11.3.9 to fit a sphere to three-dimensional data (x_i, y_i, z_i), $i = 1 : m$.

11.3 Unconstrained Optimization

11.3.1 Optimality Conditions

Consider an unconstrained optimization problem of the form

$$\min_x \phi(x), \qquad x \in \mathbf{R}^n. \tag{11.3.1}$$

where the objective function ϕ is a mapping $\mathbf{R}^n \to \mathbf{R}$. Often one would like to find a **global minimum**, i.e., a point where $\phi(x)$ assumes its least value in some subset $\times \in \mathcal{B} \subset \mathbf{R}^n$. However, this is only possible in rather special cases and most numerical methods try to find local minima of $\phi(x)$.

Definition 11.3.1.

A point x^* is said to be a **local minimum** *of ϕ if $\phi(x^*) \leq \phi(y)$ for all y in a sufficiently small neighborhood of x^*. If $\phi(x^*) < \phi(y)$ then x^* is a* **strong** *local minimum.*

Assume that the objective function ϕ is continuously differentiable at a point x with gradient vector $g(x) = \nabla\phi(x)$. The gradient vector $g(x) = \nabla\phi(x)$ is the normal to the tangent hyperplane of the multivariate function $\phi(x)$ (see Def. 11.1.3). As in the scalar case, a *necessary* condition for a point x^* to be optimal is that it satisfies the nonlinear system $g(x) = 0$.

Definition 11.3.2.

A point x^* which satisfies $g(x) = \nabla\phi(x) = 0$ is called a **stationary point**.

Definition 11.3.3.

A function $\phi : \mathbf{R}^n \to \mathbf{R}$ is twice continuously differentiable at x, if

$$g_{ij} = \frac{\partial}{\partial x_i}\left(\frac{\partial \phi}{\partial x_j}\right) = \frac{\partial^2 \phi}{\partial x_i \partial x_j}, \qquad 1 \leq i, j \leq n.$$

exist and are continuous. The square matrix $H(x)$ formed by these n^2 quantities is

called the **Hessian** *of* $\phi(x)$*,*

$$H(x) = \nabla^2\phi(x) = \begin{pmatrix} \frac{\partial^2\phi}{\partial x_1^2} & \cdots & \frac{\partial^2\phi}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2\phi}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2\phi}{\partial x_n^2} \end{pmatrix} \in \mathbf{R}^{n\times n}. \tag{11.3.2}$$

If the gradient and Hessian exist and are continuous then the Hessian matrix is *symmetric*, i.e., $\partial^2\phi/\partial x_i\partial x_j = \partial^2\phi/\partial x_j\partial x_i$. Note that information about the Hessian is needed to determine if a stationary point corresponds to a minimum of the objective function. We have the following fundamental result.

Theorem 11.3.4.

Necessary conditions for x^* *to be a local minimum of* ϕ *is that* x^* *is a stationary point, i.e.,* $g(x^*) = 0$*, and that* $H(x^*)$ *is positive semi-definite. If* $g(x^*) = 0$ *and* $H(x^*)$ *positive definite then* x^* *is a strong local minimum.*

Proof. The Taylor-series expansion of ϕ about x^* is

$$\phi(x^* + \epsilon d) = \phi(x^*) + \epsilon d^T g(x^*) + \frac{1}{2}\epsilon^2 d^T H(x^* + \epsilon\theta d)d,$$

where $0 \le \theta \le 1$, ϵ is a scalar and d a vector. Assume that $g(x^*) \neq 0$ and choose d so that $d^T g(x^*) < 0$. Then for sufficiently small $\epsilon > 0$ the last term is negligible and $\phi(x^* + \epsilon d) < \phi(x^*)$. □

Note that, as in the one-dimensional case, it is possible for a stationary point to be neither a maximum or a minimum. Such a point is called a **saddle point**, and is illustrated in Fig. 11.2.1.

Figure 11.3.1. *A saddle point.*

11.3.2 Steepest Descent

In many iterative methods for minimizing a function $\phi(x) : \mathbf{R}^n \to \mathbf{R}$, a sequence of points $\{x_k\}$, $k = 0, 1, 2, \ldots$ are generated from

$$x_{(k+1)} = x_k + \alpha_k d_k, \tag{11.3.3}$$

where d_k is a **search direction** and α_k a **step length**. If we put

$$f(\alpha) = \phi(x_k + \alpha_k d_k), \tag{11.3.4}$$

then $f'(0) = (d_k)^T g(x_k)$, where $g(x_k)$ is the gradient at x_k. The search direction d_k is said to be a **descent direction** if $(d_k)^T g(x_k) < 0$.

We assume in the following that d_k is normalized so that $\|d_k\|_2 = 1$. Then by the Schwarz inequality $f'(0)$ is minimized when

$$d_k = -g(x_k)/||g(x_k)||_2. \tag{11.3.5}$$

Hence the negative gradient direction is a *direction of steepest descent*, and this choice with $\lambda_k > 0$ leads to the **steepest descent method** (Cauchy, 1847). If combined with a suitable step length criteria this method is always guaranteed to converge to a stationary point.

In the steepest descent method the Hessian is not needed. Because of this the rate of convergence is only *linear*, and can be very slow, see Fig. 11.4.1. Hence this method is usually used only as a starting step, or when other search directions fail.

Example 11.3.1.

If the steepest descent method is applied to a quadratic function

$$\phi(x) = b^T x + \frac{1}{2} x^T G x,$$

where G is a symmetric positive definite matrix. Then from the analysis in Sec. 11.4.3 it follows that

$$\phi(x_{k+1}) - \phi(x^*) \approx \rho^2(\phi(x_k) - \phi(x^*)), \qquad \rho = \frac{\kappa - 1}{\kappa + 1},$$

where $\kappa = \kappa_2(G)$ is the condition number of G. For example, if $\kappa = 1000$, then $\rho^2 = (999/1001)^2 \approx 0.996$, and about 575 iterations would be needed to gain one decimal digit of accuracy!

11.3.3 Newton and Quasi-Newton Methods

Faster convergence can be achieved by making use, not only of the gradient, but also of the second derivatives of the objective function $\phi(x)$. The basic Newton method determines the new iterate x_{k+1}, by minimizing the **quadratic model** $\phi(x_k + s_k) \approx q_k(s_k)$,

$$q_k(s_k) = \phi(x_k) + g(x_k)^T s_k + \frac{1}{2} s_k^T H(x_k) s_k, \tag{11.3.6}$$

of the function $\phi(x)$ at the current iterate x_k. When the Hessian matrix $H(x_k)$ is positive definite, q_k has a unique minimizer that is obtained by taking $x_{k+1} = x_k + s_k$, where the **Newton step** s_k is the solution of the symmetric linear system

$$H(x_k) s_k = -g(x_k). \tag{11.3.7}$$

As in the case of solving a nonlinear system Newton's method needs to be modified when the initial point x_0 is not close to a minimizer. Either a line search can be included or a trust region technique used. In a line search we take the new iterate to be

$$x_{k+1} = x_k + \lambda_k d_k,$$

where d_k is a search direction and $\lambda_k > 0$ chosen so that $\phi(x_{k+1}) < \phi(x_k)$. The algorithms described in Section 11.3.2 for minimizing the univariate function $\phi(x_k + \lambda d_k)$ can be used to determine λ_k. However, it is usually not efficient to determine an accurate minimizer. Rather it is required that λ_k satisfy the two conditions

$$\phi(x_k + \lambda_k d_k) \leq \phi(x_k) + \mu\lambda_k g(x_k)^T d_k, \tag{11.3.8}$$

$$\left|g(x_k + \lambda_k d_k)^T d_k\right| \leq \eta \left|g(x_k)^T d_k\right|, \tag{11.3.9}$$

where μ and η are constants satisfying $0 < \mu < \eta < 1$. Typically $\mu = 0.001$ and $\eta = 0.9$ are used.

Note that the Newton step is not a descent direction if $g_k^T H(x_k)^{-1} g_k \leq 0$. This situation is not likely to occur in the vicinity of a local optimum x^*, because of the positive (or at least nonnegative) definiteness of $H(x^*)$. Far away from an optimal point, however, this can happen. This is the reason for admitting the gradient as an alternative search direction—especially since there is a danger that the Newton direction will lead to a saddle point.

In the quadratic model the term $s_k^T H(x_k) s_k$ can be interpreted as the curvature of the surface $\phi(x)$ at x_k along s_k. Often $H(x_k)$ is expensive to compute, and we want to approximate this term. Expanding the gradient function in a Taylor series about x_k along a direction s_k we have

$$g(x_k + s_k) = g_k + H(x_k)s_k + \ldots. \tag{11.3.10}$$

Hence the curvature can be approximated from the gradient using a forward difference approximation

$$s_k^T G_k s_k \approx \big(g(x_k + s_k) - g(x_k)\big)^T s_k.$$

In **quasi-Newton**, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the kth step. It is then required that B_{k+1} approximates the curvature of ϕ along $s_k = x_{k+1} - x_k$, i.e.,

$$B_{k+1}s_k = \gamma_k, \qquad \gamma_k = g(x_{k+1}) - g(x_k), \tag{11.3.11}$$

where γ_k is the change in the gradient. The first equation in (11.3.11) is the analog of the secant equation (11.3.10) and is called the **quasi-Newton condition**.

Since the Hessian matrix is symmetric, it seems natural to require also that each approximate Hessian is symmetric. The quasi-Newton condition can be satisfied by making a simple update to B_k. The Powell-Symmetric-Broyden (PSB) update is

$$B_{k+1} = B_k + \frac{r_k s_k^T + s_k r_k^T}{s_k^T s_k} - \frac{(r_k^T s_k) s_k s_k^T}{(s_k^T s_k)^2}, \tag{11.3.12}$$

where $r_k = \gamma_k - B_k s_k$. The update matrix $B_{k+1} - B_k$ is here of rank two. It can be shown that it is the unique symmetric matrix which minimizes $\|B_{k+1} - B_k\|_F$, subject to (11.3.11).

When line searches are used, practical experience has shown the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, given by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k \gamma_k^T}{\gamma_k^T s_k},$$

to be the best update.

If the choice of step length parameter λ_k is such that $\gamma_k^T s_k > 0$, then B_{k+1} will inherit positive definiteness from B_k. Therefore it is usual to combine the BFGS formula with $B_0 = I$. The most widely used algorithms for unconstrained optimization use these techniques, when it is reasonable to store B_k as a dense matrix. Note that since the search direction will be computed from

$$B_k d_k = -g_k, \qquad k \geq 1, \tag{11.3.13}$$

this means that the first iteration of a quasi-Newton method is a steepest descent step.

If B_k is positive definite then the local quadratic model has a unique local minimum, and the search direction d_k computed from (11.3.13) is a descent direction. Therefore it is usually required that the update formula generates a positive definite approximation B_{k+1} when B_k is positive definite.

To compute a new search direction we must solve the linear system (11.3.13), which in general would require order n^3 operations. However, since the approximate Hessian B_k is a rank two modification of B_{k-1}, it is possible to solve this system more efficiently. One possibility would be to maintain an approximation to the *inverse* Hessian, using the Shermann-Morrison formula (6.2.14). Then only $O(n^2)$ operations would be needed. However, if the Cholesky factorization $B_k = L_k D_k L_k^T$ is available the system (11.3.13) can also be solved in order n^2 operations. Furthermore, the factors L_{k+1} and D_{k+1} of the updated Hessian approximation B_{k+1} can be computed in about the same number of operations that would be needed to generate B_{k+1}^{-1}. An important advantage of using the Cholesky factorization is that the positive definiteness of the approximate Hessian cannot be lost through round-off errors.

An algorithm for modifying the Cholesky factors of a symmetric positive definite matrix B was given by Gill, et al. [14, 1975]. Let $B = LDL^T$ be the Cholesky factorization of B, where $L = (l_{ij})$ is unit lower triangular and $D = \operatorname{diag}(d_j) > 0$ diagonal. Let $\bar{B} = B \pm vv^T$ be a rank-one modification of B. Then we can write

$$\bar{B} = LDL^T \pm vv^T = L(D \pm pp^T)L^T,$$

where p is the solution of the triangular system $Lp = v$. The Cholesky factorization $D \pm pp^T = \hat{L}\bar{D}\hat{L}^T$ can be computed by a simple recursion, and then we have $\bar{L} = L\hat{L}$. In case of a positive correction $B = B + vv^T$, the vector p and the elements of $\bar{L}$ and $\bar{D}$ can be computed in a numerical stable way using only $3n^2/2$ flops.

Review Questions

1. Consider the unconstrained optimization problem $\min_x \phi(x)$, $x \in \mathbf{R}^n$. Give necessary conditions for x^* to be a local minimum. ($\phi(x) : \mathbf{R}^n \to \mathbf{R}$ is assumed to be twice continuously differentiable.)
2. (a) In many iterative methods for minimizing a function $\phi(x)$, a sequence of points are generated from $x_{k+1} = x_k + \lambda_k d_k$, $k = 0, 1, 2, \ldots$, where d_k is a search direction. When is d_k a descent direction? Describe some strategies to choose the step length λ_k.

   (b) Define the Newton direction. When is the Newton direction a descent direction?
3. In quasi-Newton, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the kth step. What quasi-Newton condition does B_k satisfy, and what is the geometrical significance of this condition?
4. (a) What property should the function $f(x)$ have to be unimodal in $[a, b]$?

   (b) Describe an interval reduction methods for finding the minimum of a unimodal function in $[a, b]$, which can be thought of as being analogues of the bisection method. What is its rate of convergence?

Problems

1. (a) The general form for a quadratic function is

$$\phi(x) = \frac{1}{2}x^T Gx - b^T x + c,$$

   where $G \in \mathbf{R}^{n\times n}$ is a symmetric matrix and $b \in \mathbf{R}^n$ a column vector. Show that the gradient of ϕ is $g = Gx - b$ and the Hessian is G. Also show that if $g(x^*) = 0$, then

$$\phi(x) = \phi(x^*) + \frac{1}{2}(x - x^*)^T H(x - x^*).$$

   (b) Suppose that G is symmetric and nonsingular. Using the result from (a) show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?
2. Let $\psi(x)$ be quadratic with Hessian matrix G, which need not be positive definite.
   (a) Let $\psi(\lambda) = \phi(x_0 - \lambda d)$. Show using Taylor's formula that

$$\psi(\lambda) = \psi(0) - \lambda g^T d + \frac{1}{2}\lambda^2 d^T Gd.$$

   Conclude that if $d^T Gd > 0$ for a certain vector d then $\psi(\lambda)$ is minimized when $\lambda = g^T d/d^T Gd$, and

$$\min_\lambda \psi(\lambda) = \psi(0) - \frac{1}{2}\frac{(d^T g)^2}{d^T Gd}.$$

   (b) Using the result from (a) show that if $g^T Gg > 0$ and $g^T G^{-1} g > 0$, then the steepest descent method $d = g$ with optimal λ gives a smaller reduction of ψ than Newton's method if $g^T G^{-1} g > (g^T g)^2/g^T Gg$. (The conclusion holds also if $\phi(x_0 -$

λd) can be approximated by a quadratic function of λ reasonably well in the relevant intervals.)

(c) Suppose that G is symmetric and nonsingular. Using the result from (b) show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?

11.4 Constrained Optimization

11.4.1 Introduction.

Linear optimization or **linear programming** is a mathematical theory and method of calculation for determining the minimum (or maximum) of a linear objective function, where the domain of the variables are restricted by a system of linear inequalities, and possibly also by a system of linear equations. This is famous problem which has been extensively studied since the late 1940's. Problems of this type come up, e.g., in economics, strategic planning, transportation and productions problems, telecommunications, and many other applications. Important special cases arise in approximation theory, e.g., data fitting in l_1 and l_∞ norms. The number of variables in linear optimization can be very large. Today linear programs with 5 million variables are solved!

A linear programming problem cannot be solved by setting certain partial derivatives equal to zero. As the following example shows, the deciding factor is the domain in which the variables can vary.

Example 11.4.1.

In a given factory there are three machines M_1, M_2, M_3 used in making two products P_1, P_2. One unit of P_1 occupies M_1 5 minutes, M_2 3 minutes, and M_3 4 minutes. The corresponding figures for one unit of P_2 are: M_1 1 minute, M_2 4 minutes, and M_3 3 minutes. The net profit per unit of P_1 produced is 30 dollars, and for P_2 20 dollars. What production plan gives the most profit?

Suppose that x_1 units of P_1 and x_2 units of P_2 are produced per hour. Then the problem is to maximize

$$f = 30x_1 + 20x_2$$

subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$, and

$$\begin{aligned} 5x_1 + x_2 &\leq 60 \quad \text{for } M_1, \\ 3x_1 + 4x_2 &\leq 60 \quad \text{for } M_2, \\ 4x_1 + 3x_2 &\leq 60 \quad \text{for } M_3. \end{aligned} \tag{11.4.1}$$

The problem is illustrated geometrically in Fig. 11.3.1. The first of the inequalities (11.4.2) can be interpreted that the solution (x_1, x_2) must lie on the left of or on the line AB whose equation is $5x_1 + x_2 = 60$. The other two can be interpreted in a similar way. Thus (x_1, x_2) must lie within or on the boundary of the pentagon $OABCD$. The value of the function f to be maximized is proportional to the orthogonal distance and the dashed line $f = 0$; it clearly takes on its largest value at the vertex B. Since every vertex is the intersection of two

lines, we must have equality in (at least) two of the inequalities. At the solution x^* equality holds in the inequalities for M_1 and M_3. These two constraints are called **active** at x^*; the other are **inactive**. The active constraints give two linear equations for determining the solution, $x_1 = 120/11$, $x_2 = 60/11$. Hence the maximal profit $f = 4,800/11 = 436.36$ dollars per hour is obtained b using M_1 and M_2 continuously, while M_2 is used only $600/11 = 54.55$ minutes per hour.

Figure 11.4.1. *Geometric illustration of a linear programming problem.*

11.4.2 Optimality for Inequality Constraints.

A linear programming (LP) problem can more generally be stated in the following form:

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax \geq b, \quad x \geq 0. \end{aligned} \tag{11.4.2}$$

Here $x \in \mathbf{R}^n$ is the vector of unknowns, $c \in \mathbf{R}^n$ is the **cost vector**, and $A \in \mathbf{R}^{m\times n}$ the constraint matrix. The function $c^T x$ to be minimized is called the **objective** function. (Note that the problem of maximizing $c^T x$ is equivalent to minimizing $-c^T x$.)

A single linear inequality constraint has the form $a_i^T x \geq b_i$. The corresponding equality $a_i^T x = b_i$ defines a hyperplane in $\mathbf{R}^n$. The inequality restricts x to lie on the feasible side of this hyperplane. The feasible region of the LP (11.4.2) is the set

$$\mathcal{F} = \{x \in \mathbf{R}^n \mid Ax \geq b\}. \tag{11.4.3}$$

An inequality constraint is said to be **redundant** if its removal does not alter the feasible region.

Obviously, a solution to the LP (11.4.2)can exist only if $\mathcal{F}$ is not empty. When $\mathcal{F}$ is not empty, it has the important property of being a **convex set**, which

is defined as follows: Let x and y be any two points in $\mathcal{F}$. Then the line segment

$$\{z \equiv (1-\alpha)x + \alpha y \mid 0 \le \alpha \le 1\}$$

joining x and y is also in $\mathcal{F}$. It is simple to verify that $\mathcal{F}$ defined by (11.4.3) has this property, since

$$Az = (1-\alpha)Ax + \alpha Ay \ge (1-\alpha)b + \alpha b = b.$$

when $0 \le \alpha \le 1\}$.

The **active set** of the inequality constraints $Ax \ge b$ at a point x is the subset of constraints which are satisfied with equality at x. Hence the constraint $a_i^T x \ge b_i$ is active if the residual at x is zero,

$$r_i(x) = a_i^T x - b_i = 0.$$

Let x be a feasible point in $\mathcal{F}$. Then it is of interest to find directions p such that $x + \alpha p$ remains feasible for some $\alpha > 0$. If the constraint $a_i^T x \ge b_i$ is active at x, then all points $y = x + \alpha p$, $\alpha > 0$, will remain feasible with respect to this constraint if and only if $a_i^T p \ge 0$. It is not difficult to see that the feasible directions p are not affected by the inactive constraints at x. Hence p is a feasible directions at the point x if and only if $a_i^T p \ge 0$ for all active constraints at x.

Given a feasible point x the maximum step α that can be taken along a feasible direction p depends on the inactive constraints. We need to consider the set of inactive constraints i for which $a_i^T p < 0$. For these constraints, which are called decreasing constraints, $a_i^T(x+\alpha p) = a_i^T x + \alpha a_i^T p = b_i$, and thus the constraint i becomes active when

$$\alpha = \alpha_i = \frac{a_i^T x - b_i}{-a_i^T p}.$$

Hence the largest step we can take along p is $\max \alpha_i$ where we maximize over all decreasing constraints.

For an LP there are three possibilities: There may be no feasible points, in which case the LP has no solution; there may be a feasible point x^* at which the objective function is minimized; Finally, the feasible region may be unbounded and the objective function unbounded below in the feasible region. The following fundamental theorem states how these three possibilities can be distinguished:

Theorem 11.4.1.

Consider the linear program minimizing $c^T x$ subject to $Ax \ge b$. (We assume here that the constraints $x \ge 0$ are not present.) Then the following results hold:

(a) *If no points satisfy $Ax \ge b$, the LP has no solution;*

(b) *If there exists a point x^* satisfying the conditions*

$$Ax^* \ge b, \qquad c = A_A^T \lambda_A^*, \quad \lambda_A^* \ge 0,$$

where A_A is the matrix of active constraints at x^*, then $c^T x^*$ is the unique minimum value of $c^T x$ in the feasible region, and x^* is a minimizer.

(c) *If the constraints $Ax \geq b$ are consistent, the objective function is unbounded below in the feasible region if and only if the last two conditions in* (b) *are not satisfied at any feasible point.*

The last two conditions in (b) state that c can be written as a nonnegative linear combination of the rows in A corresponding to the active constraints. The proof of this theorem is nontrivial. It is usually proved by invoking **Farkas Ĺemma**, a classical result published in 1902. For a proof we refer to [17, Sec. 7.7].

The geometrical ideas in the introductory example are useful also in the general case. Given a set of linear constraints a **vertex** is a feasible point for which the active constraints matrix has rank n. Thus at least n constraints are active at a vertex x. A vertex is an extreme point of the feasible region $\mathcal{F}$. If exactly n constraints are active at a vertex, the vertex is said to be **nondegenerate**; if more than n constraints are active at a vertex, the vertex is said to be **degenerate**. In Example 11.4.1 there are five vertices O, A, B, C, and D, all of which are nondegenerate. The vertices form a polyhedron, or **simplex** in $\mathbf{R}^n$.

Vertices are of central importance in linear programming since many LP have the property that a minimizer lies at a vertex. The following theorem states the conditions under which this is true.

Theorem 11.4.2.
Consider the linear program of minimizing $c^T x$ subject to $Ax \geq b$, $A \in \mathbf{R}^{m\times n}$. If $\operatorname{rank}(A) = n$ and the optimal value of $c^T x$ is finite, a vertex minimizer exists.

Note that by convexity an infinity of non-vertex solutions will exist if the minimizer is not unique. For example, in a problem like Example 11.4.1, one could have an objective function $f = c^T x$ such that the line $f = 0$ were parallel to one of the sides of the pentagon. Then all points on the line segment between two optimal vertices in the polyhedron are also optimal points.

Suppose a linear program includes the constraints $x \geq 0$. Then the constraint matrix has the form

$$\begin{pmatrix} A \\ I_n \end{pmatrix} \in \mathbf{R}^{(m+n)\times n}.$$

Since the rows include the identity matrix I_n this matrix always has rank n. Hence a feasible vertex must exist if any feasible point exists.

11.4.3 Standard Form LP.

It is convenient to adopt the following **standard form** of a linear programming problem:

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax = b, \quad x \geq 0. \end{aligned} \tag{11.4.4}$$

where $A \in \mathbf{R}^{m\times n}$. The constraints $x \geq 0$ are the only inequality constraints in a standard form problem. The set $\mathcal{F}$ of feasible points consists of points x that satisfy

$Ax = b$ and $x \geq 0$. If $\operatorname{rank}(A) = n$ this set contains just one point if $A^{-1}b \geq 0$; otherwise it is empty. Hence in general we have $\operatorname{rank}(A) < n$.

It is simple to convert a linear programming problem to standard form. Many LP software packages apply an automatic internal conversion to standard form. The change of form involves modification of the dimensions, variables and constraints. An upper bound inequality $a^T x \leq \beta$ is converted into an equality $a^T x + s = \beta$ by introducing a **slack variable** s subject to $s \geq 0$. A lower bound inequality of the form $a^T x \geq \beta$ can be changed to an upper bound inequality $(-a)^T x \leq -\beta$. Thus when a linear programming problems with inequality constraints is converted to standard form, the number of variables will increase. If the original constraints are $Ax \leq b$, $A \in \mathbf{R}^{m\times n}$, then the matrix in the equivalent standard form will be $(\, A \quad I_m \,)$, and the number of variables n plus m slack variables.

Example 11.4.2.

The problem in Example 11.4.1 can be brought into standard form with the help of three slack variables, x_3, x_4, x_5. We get

$$A = \begin{pmatrix} 5 & 1 & 1 & & \\ 3 & 4 & & 1 & \\ 4 & 3 & & & 1 \end{pmatrix}, \quad b = 60 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$
$$c^T = (\, -20 \quad -30 \quad 0 \quad 0 \quad 0 \,).$$

The three equations $Ax = b$ define a two-dimensional subspace (the plane in Fig. 11.4.1) in the five-dimensional space of x. Each side of the pentagon $OABCD$ has an equation of the form $x_i = 0$, $i = 1 : 5$. At a vertex two of the coordinates are zero, and the rest cannot be negative.

For completeness we note that, although this is seldom used in practice, equality constraints can be converted to inequality constraints. For example, $a_i^T x = b_i$ is equivalent to the two inequality constraints $a_i^T x > b_i$ and $-a_i^T x \geq -b_i$.

The optimality conditions for an LP in standard form are as follows:

Theorem 11.4.3.

Consider the standard linear program of minimizing $c^T x$ subject to $Ax = b$ and $x \geq 0$ for which feasible points exist. Then x^* is a minimizer if and only if x^* is a feasible point and

$$c = A^T \pi^* + \eta^*, \qquad \eta^* \geq 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \ldots, n. \tag{11.4.5}$$

A vertex for a standard form problem is also called a **basic feasible point**. In case more than $n - m$ coordinates are zero at a feasible point we say that the point is a **degenerate** feasible point. A feasible vertex must exist if any feasible point exists. Since the m equality constraints are active at all feasible points, at least $n - m$ of the bound constraints must also be active at a vertex. It follows that a point x can be a vertex only if at least $n - m$ of its components are zero.

In the following we assume that there exist feasible points, and that $c^T x$ has a finite minimum. Then an eventual unboundedness of the polyhedron does not give rise to difficulties. These assumptions are as a rule satisfied in all practical problems which are properly formulated.

We have the following fundamental theorem, the validity of which the reader can easily convince himself of for $n - m \leq 3$.

Theorem 11.4.4.

For a linear programming problem in standard form some optimal feasible point is also a basic feasible point, i.e., at least $n - m$ of its coordinates are zero; equivalently at most m coordinates are strictly positive.

The standard form given above has the drawback that when variables are subject to lower and upper bounds, these bounds have to be entered as general constraints in the matrix A. Since lower and upper bounds on x can be handled much more easily, a more efficient formulation is often used where inequalities $l \leq x \leq u$ are substituted for $x \geq 0$. Then it is convenient to allow $l_i = -\infty$ and $u_i = \infty$ for some of the variables x_i. If for some j, $l_j = -\infty$ and $u_j = \infty$, x_j is said to be free, and if for some j, $l_j = u_j$, x_j is said to be fixed. For simplicity we consider in the following mainly the first standard form.

Example 11.4.3.

As a nontrivial example of the use of Theorem 11.4.2 we consider the following **transportation problem**, which is one of the most well-known problems in optimization. Suppose that a business concern has I factories which produce $a_1, a_2, \ldots, a_I$ units of a certain product. This product is sent to J consumers, who need $b_1, b_2, \ldots, b_J$ units, respectively. We assume that the total number of units produced is equal to the total need, i.e., $\sum_{i=1}^{I} a_i = \sum_{j=1}^{J} b_j$. The cost to transport one unit from producer i to consumer j equals c_{ij}. The problem is to determine the quantities x_{ij} transported so that the total cost is minimized. This problem can be formulated as a linear programming problem as follows:

$$\text{minimize} \quad f = \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij}$$

subject to $x_{ij} \geq 0$, and the constraints

$$\sum_{j=1}^{J} x_{ij} = a_i, \quad i = 1 : I, \qquad \sum_{i=1}^{I} x_{ij} = b_j, \quad j = 1 : J.$$

There is a linear dependence between these equations, since

$$\sum_{i=1}^{I} \sum_{j=1}^{J} x_{ij} - \sum_{j=1}^{J} \sum_{i=1}^{I} x_{ij} = 0.$$

The number of linearly independent equations is thus (at most) equal to $m = I + J - 1$. From Theorem 11.4.2 it follows that *there exist an optimal transportation*

scheme, where at most $I + J - 1$ of the IJ possible routes between producer and consumer are used. In principle the transportation problem can be solved by the simplex method described below; however, there are much more efficient methods which make use of the special structure of the equations.

Many other problems can be formulated as transportation problems. One important example is the **personnel-assignment problem**: One wants to distribute I applicants to J jobs, where the suitability of applicant i for job j is known. The problem to maximize the total suitability is clearly analogous to the transportation problem.

11.4.4 The Simplex Method

The **simplex method** was invented in 1947 by G. B. Danzig. Until the late 1980s it was the only effective algorithm for solving large linear programming problems. Later the simplex method has been rivaled by so called interior-point methods (see Section 11.4.7), but it is still competitive for many classes of problems.

The idea behind the simplex method is simple. From Theorem 11.4.2 we know that the problem is solved if we can find out which of the n coordinates x are zero at the optimal feasible point. In theory, one could consider trying all the $\binom{n}{n-m}$ possible ways of setting $n-m$ variables equal to zero, sorting out those combinations which do not give feasible points. The rest are vertices of the polyhedron, and one can look among these to find a vertex at which f is minimized. However, since the number of vertices increases exponentially with $n-m$ this is laborious even for small values of m and n.

The simplex method starts at a vertex (basic feasible point) and recursively proceeds from one vertex to an adjacent vertex with a lower value of the objective function $c^T x$. The first phase in the simplex method is to determine an initial basic feasible point (vertex). In some cases an initial vertex can be trivially found (see, e.g., Example 11.4.4 below). A systematic method which can be used in more difficult situations will be described later in Section 11.4.5.

When an initial feasible point has been found, the following steps are repeated until convergence:

I. Check if the current vertex is an optimal solution. If so then stop, else continue.

II. Proceed from the current vertex to a neighboring vertex at which the value of f if possible is smaller.

Consider the standard form linear programming problem (11.4.4). At a vertex $n-m$ variables are zero. We divide the index set $I = \{1 : m\}$ into two disjoint sets

$$I = B \cup N, \quad B = \{j_1, \ldots, j_n\}, \quad N = \{i_1, \ldots, i_{n-m}\}, \tag{11.4.6}$$

such that N corresponds to the zero variables. We call x_B **basic variables** and x_N **nonbasic variables**. If the vector x and the columns of the matrix A are split in

a corresponding way, we can write the system $Ax = b$ as

$$A_B x_B = b - A_N x_N. \tag{11.4.7}$$

We start by illustrating the simplex method on the small example from the introduction.

Example 11.4.4.

In Example 11.4.2 we get an initial feasible point by taking $x_B = (x_3, x_4, x_5)^T$ and $x_N = (x_1, x_2)^T$. The corresponding splitting of A is

$$A_B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad A_N = \begin{pmatrix} 5 & 1 \\ 3 & 4 \\ 4 & 3 \end{pmatrix},$$

Putting $x_N = 0$ gives $\hat{x}_B = b = (60, 60, 60)^T$. Since $x_B \geq 0$ this corresponds to a vertex (the vertex O in Fig. 11.4.1) for which $f = 0$. The optimality criterion is not fulfilled since $c_B = 0$ and $\hat{c}_N^T = c_N^T = (-30, -20) < 0$. If we choose $x_r = x_1 = \theta > 0$ then using $\hat{x}_B$ and the first column of $A_B^{-1} A_N = A_N$ we find

$$\theta_{max} = 60 \min_i \{1/5, 1/3, 1/4\} = 12.$$

Clearly a further increase in x_1 is inhibited by x_3. We now exchange these variables to get $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)^T$. (Geometrically this means that one goes from O to A in Fig. 11.4.1.)

The new sets of basic and non-basic variables are $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)$. Taking $x_N = 0$ we have

$$\hat{x}_B = (12, 24, 12)^T, \qquad f = 0 + 12 \cdot 30 = 360.$$

The new splitting of A is

$$A_B = \begin{pmatrix} 5 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}, \qquad A_N = \begin{pmatrix} 1 & 1 \\ 0 & 4 \\ 0 & 3 \end{pmatrix}.$$

The reduced costs $\hat{c}_N^T = (\,6 \quad -14\,)$ for non-basic variables are easily computed from (11.4.10). The optimality criterion is not satisfied. We take $x_r = x_2$ and solve $A_B b_2 = a_2$ to get $b_2 = (1/5)(1, 17, 11)^T$. We find $\theta = 5 \min(12/1, 24/17, 12/11) = 60/11$. Exchanging x_2 and x_5 we go from A to B in Fig. 11.4.1. The new basic variables are

$$\hat{x}_B = (x_1, x_4, x_2) = (5/11)(24, 12, 12)^T, \qquad f = 4,800/11.$$

The non-basic variables are $x_N = (x_3, x_5)^T$, and to compute the reduced costs we must solve

$$\begin{pmatrix} 5 & 3 & 4 \\ 0 & 1 & 0 \\ 1 & 4 & 3 \end{pmatrix} d = \begin{pmatrix} -30 \\ 0 \\ -20 \end{pmatrix}.$$

We have $c_N^T = (0, 0)$ and get $\hat{c}_N^T = (d_1, d_3) = \frac{10}{11}\,(\,1 \quad 7\,)$. The optimality criterion is now satisfied, so we have found the optimal solution.

In older textbooks the calculations in the simplex method is usually presented in form of a **tableau**, where the whole matrix $B_N = A_B^{-1}A_N$ is updated in each step; see Problem 3. However, these formulas are costly and potentially unstable, since they do not allow for pivoting for size.

We now give a general description of the steps in the simplex method which is closer to what is used in current simplex codes. We assume that the matrix A_B in (11.4.7) is nonsingular. (This will always be the case if rank $(A) = n$.) We can then express the basic variables in in terms of the nonbasic

$$x_B = \hat{x}_B - A_B^{-1}A_N x_N, \quad \hat{x}_B = A_B^{-1}b, \tag{11.4.8}$$

where $\hat{x}_B$ is obtained by solving the linear system $A_B\hat{x}_B = b$. If $\hat{x}_B \geq 0$ then $x_N = 0$ corresponds to a basic feasible point (vertex). The vector c is also split in two subvectors c_B and c_N, and using (11.4.8) we have

$$f = c^T x = c_B^T(\hat{x}_B - A_B^{-1}A_N x_N) + c_N^T x_N = c_B^T\hat{x}_B + \hat{c}_N^T x_N,$$

where

$$\hat{c}_N = c_N - A_N^T d, \qquad d = A_B^{-T}c_B. \tag{11.4.9}$$

Here d can be computed by solving the linear system

$$A_B^T d = c_B. \tag{11.4.10}$$

The components of $\hat{c}_N$ are known as the **reduced costs** for the nonbasic variables, and the process of computing them known as **pricing**. If $\hat{c}_N \geq 0$ *then* $x_N = 0$ *corresponds to an optimal point, since* f *cannot decrease when one gives one (or more) nonbasic variables positive values* (negative values are not permitted). Hence if the **optimality criterion** $\hat{c}_N \geq 0$ is satisfied, then the solution $x_B = \hat{x}_B$, $x_N = 0$ is optimal, and we can stop.

If the optimality criterion is *not* satisfied, then there is at least one non-basic variable x_r whose coefficient $\hat{c}_r$ in $\hat{c}_N$ is negative. We now determine the largest positive increment one can give x_r without making any of the basic variables negative, while holding the other non-basic variables equal to zero. Consider equation (11.4.8), and let b_r be the corresponding column of the matrix $A_B^{-1}A_N$. This column can be determined by solving the linear system

$$A_B b_r = a_r, \tag{11.4.11}$$

where a_r is the column in A_N corresponding to x_r. If we take $x_r = \theta > 0$, then $x_B = \hat{x}_B - \theta b_r$, and for any basic variable x_i we have $x_i = \hat{x}_i - \theta b_{ir}$. Hence if $b_{ir} > 0$, then x_i remains positive for $\theta = \theta_i \leq \hat{x}_i/b_{ir}$. The largest θ for which no basic variable becomes negative is given by

$$\theta = \min_i \theta_i, \qquad \theta_i = \begin{cases} \hat{x}_i/b_{ir} & \text{if } b_{ir} > 0; \\ +\infty & \text{if } b_{ir} \leq 0; \end{cases} \tag{11.4.12}$$

If $\theta = +\infty$, the object function is unbounded in the feasible region, and we stop. Otherwise there is at least one basic variable x_l that becomes zero for this value of θ. Such a variable is now interchanged with x_r, so x_r becomes a basic variable and x_l a non-basic variable. (Geometrically this corresponds to going to a neighboring vertex.) Note that the new values of the basic variables can easily be found by updating the old values using $x_i = x_i - \theta b_{ir}$, and $x_r = \theta$.

In case several components of the vector $\hat{c}_N$ are negative we have to specify which variable to choose. The so-called **textbook** strategy chooses r as the index of the most negative component in $\hat{c}_N$. This can be motivated by noting that c_r equals the reduction in the object function $f = c_B^T \hat{x}_B + \hat{c}_N^T x_N$, produced by a unit step along x_r Hence this choice leads to the largest reduction in the objective function assuming a fixed length of the step. A defect of this strategy is that it is not invariant under scalings of the matrix A. A scaling invariant strategy called the **steepest edge strategy** can lead to great gains in efficiency, see Gill, Murray, and Wright [17, 1991, Ch. 8].

It is possible that even at a vertex which is not an optimal solution one cannot increase f by exchanging a single variable without coming in conflict with the constraints. This exceptional case occurs only when one of the basic variables is zero at the same time that the non-basic variables are zero. As mentioned previously such a point is called a **degenerate vertex**. In such a case one has to exchange a non-basic variable with one of the basic variables which is zero at the vertex, and a step with $\theta = 0$ occurs. In more difficult cases, it may even be possible to make several such exchanges.

Figure 11.4.2. *Feasible points in a degenerate case.*

Example 11.4.5.

Suppose we want to maximize $f = 2x_1 + 2x_2 + 3x_3$ subject to the constraints

$$x_1 + x_3 \le 1, \quad x_2 + x_3 \le 1, \quad x_i \ge 0, \quad i = 1, 2, 3.$$

The feasible points form a four-sided pyramid in (x_1, x_2, x_3)-space; see Fig. 11.3.2. Introduce slack variables x_4 and x_5, and take $\{x_1, x_2, x_3\}$ as non-basic variables.

This gives a feasible point since $x_1 = x_2 = x_3 = 0$ (the point O in Fig. 11.3.2) satisfies the constraints. Suppose at the next step we move to point A, by exchanging x_3 and x_4. At this point the non-basic variables $\{x_1, x_2, x_4\}$ are zero but also x_5, and A is a degenerate vertex, and we have

$$\begin{aligned} x_3 &= 1 - x_1 - x_4, \\ x_5 &= x_1 - x_2 + x_4, \\ f &= 3 - x_1 + 2x_2 - 3x_4. \end{aligned}$$

The optimality condition is not satisfied, and at the next step we have to exchange x_2 and x_5, and remain at point A. In the final step we can now exchange x_1 and x_2 to get to the point B, at which

$$\begin{aligned} x_1 &= 1 - x_3 - x_4, \\ x_2 &= 1 - x_3 - x_5, \\ f &= 4 - x_3 - x_4 - 2x_5. \end{aligned}$$

The optimality criterion is fulfilled, and so B is the optimal point.

Traditionally degeneracy has been a major problem with the Simplex method. A proof that the simplex algorithm converges after a finite number of steps relies on a strict increase of the objective function in each step. When steps in which f does not increase occur in the simplex algorithm, there is a danger of **cycling**, i.e., the same sequence of vertices are repeated infinitely often, which leads to non-convergence. Techniques exist which prevent cycling by allowing slightly infeasible points, see Gill, Murray and Wright [17, 1991, Sec. 8.3.3]. By perturbing each bound by a small random amount, the possibility of a tie in choosing the variable to leave the basis is virtually eliminated.

Most of the computation in a simplex iteration is spent with the solution of the two systems of equations $A_B^T d = c_B$ and $A_B b_r = a_r$. We note that both the matrix A_B and the right hand sides c_B and a_r are often very sparse. In the original simplex method these systems were solved by recurring the inverse A_B^{-1} of the basis matrix. However, this is in general inadvisable because of lack of numerical stability.

Stable methods can be devised which store and update the LU factorization

$$P_B A_B = LU \tag{11.4.13}$$

where P_B is a permutation matrix. The initial factorization (11.4.13) is computed by Gaussian elimination and partial pivoting. The new basis matrix which results from dropping the column a_r and inserting the column a_s in the last position is a Hessenberg matrix. Special methods can therefore be used to generate the factors of the subsequent basis matrices as columns enter or leave.

From the above it is clear that the major computational effort in a simplex step is the solution of the two linear systems

$$A_B^T \hat{d} = c_B, \qquad A_B b_r = a_r, \tag{11.4.14}$$

to compute reduced costs and update the basic solution. These systems can be solved cheaply by computing a LU factorization of the matrix A_B is available. For large problems it is essential to take advantage of sparsity in A_B. In particular the initial basis should be chosen such that A_B has a structure close to diagonal or triangular. Therefore row and column permutations are used to bring A_B into such a form. Assume that a LU factorization has been computed for the initial basis. Since in each step only *one column* in A_B is changed, techniques for updating a (sparse) LU factorization play a central role in modern implementation of the simplex method.

Although the worst case behaviour of the simplex method is very poor–the number of iterations may be exponential in the number of unknowns—this is never observed in practice. Computational experience indicates that the simplex methods tends to give the exact result after about $2m$–$3m$ steps, and essentially independent of the number of variables n. Note that the number of iterations can be decreased substantially if one starts from an initial point close to an optimal feasible point. In some cases it may be possible to start from the optimal solution of a nearby problem. (This is sometimes called "a warm start".)

11.4.5 Finding an Initial Basis

It may not be trivial to decide if a feasible point exists, and if so, how to find one. Modify the problem by introducing a sufficient number of new **artificial variables** are added to the constraints in (11.4.4) to assure that an initial bases matrix A_B can be found satisfying

$$x_B = A_B^{-1} b \geq 0, \qquad x_N = 0, \qquad x = (x_B^T, x_N^T)^T.$$

By introducing large positive costs associated with the artificial variables these are driven towards zero in the initial phase of the Simplex algorithm. If a feasible point exists, then eventually all artificial variables will become non-basic variables and can be dropped. This is often called the **phase 1** in the solution of the original linear program. The following example illustrates this technique.

Example 11.4.6.

Maximize $f = x_1 - x_2$, subject to the constraints $x_i \geq 0$, $i = 1:5$, and

$$\begin{aligned} x_3 &= -2 + 2x_1 - x_2, \\ x_4 &= 2 - x_1 + 2x_2, \\ x_5 &= 5 - x_1 - x_2. \end{aligned}$$

Here if $x_1 = x_2 = 0$, x_3 is negative, so x_1, x_2 cannot be used as non-basic variables. It is not immediately obvious which pair of variables suffice as non-basic variables. Modify the problem by introducing a new **artificial variable** $x_6 \geq 0$, defined by the equation

$$x_6 = 2 - 2x_1 + x_2 + x_3.$$

We can now take x_4, x_5, x_6 as basic variables, and have found a feasible point for an extended problem with six variables. This problem will have the same solution as

the original, if we can ensure that the artificial variable x_6 is zero at the solution. To accomplish this we modify the objective function to become

$$\bar{f} = x_1 - x_2 - Mx_6 = -2M + (1+2M)x_1 - (1+M)x_2 - Mx_3.$$

Here M is assumed to be a large positive number, much larger than other numbers in the computation. Then a positive value of x_6 will tend to make the function to be maximized quite small, which forces the artificial variable to become zero at the solution. Indeed, as soon as x_6 appears as a nonbasic variable, (this will happen if x_1 and x_6 are exchanged here) it is no longer needed in the computation, and can be deleted, since we have found an initial feasible point for the original problem.

The technique sketched above may be quite inefficient. A significant amount of time may be spent minimizing the sum of the artificial variables, and may lead to a vertex far away from optimality. We note that it is desirable to choose the initial basis so that A_B has a diagonal or triangular structure. Several such basis selection algorithms, named basis crashes, have been developed, see Bixby [2, 1992].

11.4.6 Duality

Consider the linear programming problem in standard form

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax = b, \quad x \geq 0. \end{aligned}$$

When this problem has a bounded optimal minimizer x^* The optimality conditions of Theorem 11.4.3 imply the existence of Lagrange multipliers y^* such that

$$c = A^T y^* + \eta^*, \qquad \eta^* \geq 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \ldots, n.$$

It follows that y^* satisfies the inequality constraints $y^T A \leq c^T$. This leads us to define the **dual problem** to the standard form problem as follows:

$$\begin{aligned} &\max_{y\in\mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad y^T A \leq c^T. \end{aligned} \tag{11.4.15}$$

Here y are the **dual variables**. The initial problem will be called the **primal problem** and x the **primal variables**. If y satisfies the inequality in (11.4.15) y is called a feasible point of the dual problem. Note that the constraint matrix for the dual problem is the transposed constraint matrix of the primal, the right-hand side in the dual is the normal vector of the primal objective, and the normal vector of the dual objective is the right-hand side of the primal.

Note that the dual to a standard form linear programming problem is in all inequality form. However, the dual problem may also be written in standard form

$$\begin{aligned} &\max_{y\in\mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad A^T y + z = c, \qquad z \geq 0, \end{aligned} \tag{11.4.16}$$

where z are the dual slack variables. The solution y^* to the dual problem is the Lagrange multiplier for the m linear equality constraints in the primal problem. The primal solution x^* is the Lagrange multiplier for the n linear equality constraints of the standard-form dual problem.

Let x and y be arbitrary feasible vectors for the primal and dual problems, respectively. Then

$$g(y) = y^T b = y^T Ax \leq c^T x = f(x). \tag{11.4.17}$$

The nonnegative quantity

$$c^T x - y^T b = x^T z$$

is called the **duality gap**. We will show it is zero if and only if x and y are optimal for the primal and dual.

Theorem 11.4.5.

The optimal values of the primal and dual problem are equal, i.e.,

$$\max g(y) = \min f(x). \tag{11.4.18}$$

The minimum value is obtained at a point $\hat{y}$ which is the solution of the m simultaneous equations

$$\hat{y}^T a_i = c_i, \quad i \in S, \tag{11.4.19}$$

where the set S is the set of integers defined previously.

Proof. By (11.4.17) it holds that

$$\max g(y) \leq \min f(x). \tag{11.4.20}$$

We shall show that $\hat{y}$ as defined by (11.4.19) is a feasible vector. Since $Ax = b$, we may write

$$f(x) = c^T x - \hat{y}^T(Ax - b) = \hat{y}^T b + (c^T - \hat{y}^T A)x.$$

Hence by (11.4.19)

$$f(x) = y^T b + \sum_{j \notin S} (c_j - \hat{y}^T a_j) x_j. \tag{11.4.21}$$

Now $f(x)$ is expressed in terms of the nonbasic variables corresponding to the optimal solution of the primal. It then follows form the optimality criterion (see Section 11.4.4) that $c_j - \hat{y}^T a_j \geq 0$, $j \notin S$. This together with (11.4.19), shows that $\hat{y}$ is a feasible point for the dual. Moreover, since $\hat{x}_j = 0$, $j \notin S$, then by (11.4.21) $f(\hat{x}) = \hat{y}^T b = g(\hat{y})$. This is consistent with (11.4.20) only if $\max g(y) = g(\hat{y})$. Hence $\max g(y) = \min f(x)$, and the theorem is proved. □

A linear program initially given in the inequality form (11.4.15)–(11.4.17) can be converted to standard form by adding n slack variables. If the simplex method is used to solve this standard problem, each step involves solution of a linear system of sizes $n \times n$. If n is large it may be advantageous to switch instead to the primal problem, which is already in standard form. A simplex step for this problem involves solving linear systems of size $m \times m$, which may be much smaller size!

11.4.7 Interior Point Methods

Interior point methods for optimization problems were introduced by Fiacco and McCormick [1968]. They are characterized by the property that a sequence of approximation strictly inside the feasible region are generated. They work by introducing a nonlinear barrier function which makes the approximations stay away from the boundary. Interest in interior point methods for linear programming did not arise until the work by Karmarkar [20, 1984], since solving a *linear* problem by techniques from *nonlinear* optimization was not believed to be a competitive approach. Currently the best interior point methods are competitive with the Simplex method on most problems and superior for some.

The most promising interior point method for linear programming is the so called primal-dual logarithmic barrier method, which we sketch below. We add a logarithmic barrier to the dual problem (11.4.15) and consider

$$\begin{aligned} &\text{maximize} \quad g = y^T b + \mu \sum_{j=1}^{n} \ln z_j, \\ &\text{subject to} \quad y^T A \le c^T. \end{aligned} \tag{11.4.22}$$

The first order optimality conditions for (11.4.22) can be shown to be

$$\begin{aligned} XZe &= \mu e, \\ Ax &= b \\ A^T y + z &= c, \end{aligned} \tag{11.4.23}$$

where $e = (1, 1, \ldots, 1)^T$, and $X = \operatorname{diag}(x)$, $Z = \operatorname{diag}(z)$. Let $\mu > 0$ be a parameter (which we will let tend to zero). Note that the last two sets of equations are the primal and dual feasibility equations, and in the limit $\mu \to 0$ the first set of equations expresses the complementarity condition $y^T x = 0$.

We then have a set of (partly nonlinear) equations for the unknown variables x, y, z. If we apply Newton's method the corrections will satisfy the following system of linear equations

$$\begin{aligned} Z\delta x + X\delta z &= \mu e - XZe, \\ A\delta x &= b - Ax, \\ A^T\delta y + \delta z &= c - A^T y - z. \end{aligned} \tag{11.4.24}$$

If $Z > 0$ we can solve to get

$$\begin{aligned} AZ^{-1}XA^T\delta y &= -AZ^{-1}(\mu e - XZe) + AZ^{-1}r_D + r_P, \\ \delta z &= -A^T\delta y + r_D, \\ \delta x &= Z^{-1}(\mu e - XZe) - Z^{-1}X\delta z. \end{aligned}$$

A sparse Cholesky factorization of ADA^T, where $D = Z^{-1}X$ is a positive diagonal matrix, is the main computational cost for the solution. Note that we need not worry about feasibility. The idea is to follow a **central path** $y(\mu)$ when $\mu \to 0$.

Review Questions

1. Give the standard form for a linear programming problem. Define the terms feasible point, basic feasible point, and slack variable.
2. State the basic theorem of linear optimization (Theorem 11.4.2). Can there be more than one optimal solution? Is every optimal solution a basic feasible point?
3. Describe the simplex method. What does one do in the case of a degenerate feasible vector?
4. Give the dual problem to $\min c^T x$ subject $Ax = b$, $x \geq 0$. How are the solutions to the dual and primal problems related?

Problems

1. (a) Find the maximum of $f = x_1 + 3x_2$ subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$,

$$x_1 + x_2 \leq 2, \qquad x_1 + 2x_2 \leq 2.$$

First solve the problem graphically, and then use the simplex method with $x_1 = x_2 = 0$ as initial point. In the following variants, begin at the optimal vertex found in problem (a).

(b) Find the maximum of $f = 2x_1 + 5x_2$ under the same constraints as in (a).

(c) Find the maximum of $f = x_1 + x_2$ under the same constraints as in (a).

(d) Find the maximum of $f = x_1 + 3x_2$ after changing the second constraint in (a) to $2x_1 + 2x_2 \leq 3$.

2. Suppose that there is a set of programs LP that solves linear programming problems in standard form. One wants to treat the problem to minimize $f = d^T x$, $d^T = (1, 2, 3, 4, 5, 1, 1)$, where $x_i \geq 0$, $i = 1 : 7$,

$$\begin{aligned} |x_1 + x_2 + x_3 - 4| &\leq 12 \\ 3x_1 + x_2 + 5x_4 &\leq 6 \\ x_1 + x_2 + 3x_3 &\geq 3 \\ |x_1 - x_2 + 5x_7| &\geq 1 \end{aligned}$$

Give A, b, and c in the standard form formulation in this case.

3. At each stage in the simplex method a basic variable x_l is exchanged with a certain nonbasic variable x_r. Before the change we have for each basic variable x_i a linear relation

$$x_i = b_{ir} x_r + \sum b_{ik} x_k, \qquad i \in L,$$

where the sum is taken over all nonbasic variables except x_r. If the equation for $i = l$ is used to solve for x_r we get

$$x_r = \frac{1}{b_{lr}} x_l - \sum \frac{b_{lk}}{b_{lr}} x_k.$$

If this expression is substituted in the rest of the equations we obtain after the exchange a relation of the form

$$x_i = \hat{b}_{il} x_l + \sum \hat{b}_{ik} x_k, \qquad i \neq l,$$

(even for $i = r$), where the sum is now taken over all the nonbasic variables except x_l. Express the coefficients in the new relation in terms of the old coefficients.

4. (a) Put the dual problem in normal form, defined in Section 11.4.3. (Note that there is no non-negativity condition on y.)

(b) Show that the dual problem of the dual problem is the primal problem.

Notes and References

The numerical solution of nonlinear equations by the methods of Newton, Brown and Brent is discussed by Moré and Cosnard [27, 1979]. An evaluation of numerical software that solves systems of nonlinear equations is given by Hiebert [19, 1982]. Here eight different available Fortran codes are compared on a set of test problems. Of these one uses a quasi-Newton method, two Brown's method, one Brent's method, and the remaining four Powell's hybrid method, see Powell [31, 1970]. A standard treatment of continuation methods is Allgower and Georg [1, 1990].

As the name implies the Gauss–Newton method was used already by Gauss [1809].

A standard textbook on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares is Dennis and Schnabel [10, 1983].

A classical reference on linear optimization is Danzig [7, 1965]. For nonlinear optimization the book by Luenberger [25, 1979] is a good introduction. Excellent textbooks on optimization are Gill, Murray and Wright [16, 1981], and Fletcher [12, 1987]. A very useful reference on software packages for large scale optimization is Moré and Wright [28, 1993] For difficult optimization problems where the gradient is not available direct search methods may be an the only option. An excellent review of this class of methods is given in [22].

A review of developments and applications of the variable projection approach for separable nonlinear least squares problems is given by Golub and Pereyra [18].

[1] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer Series in Computational Mathematics, Vol. 13. Springer-Verlag, Berlin, 1990.

[2] R. E. Bixby. Implementing the simplex method: The initial basis. *ORSA J. Comput.*, 4:267–284, 1992.

[3] P. T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Statist. Comput.*, 8:1052–1078, 1987.

[4] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[5] Y. Censor and S. A. Zenios. *Parallel Optimization. Theory, Algorithms, and Applications*. Oxford University Press, Oxford, 1997.

[6] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.

[7] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, second edition, 1965.

[8] J. E. Dennis, D. M. Gay, and R. E. Welsh. An adaptive nonlinear least-squares algorithm. *ACM. Trans. Math. Software*, 7:348–368, 1981.

[9] J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977.

[10] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics 16. SIAM, Philadelphia, PA, 1995.

[11] J. Dieudonné. *Foundations of Modern Analysis.* Academic Press, New York, NY, 1961.

[12] R. Fletcher. *Practical Methods of Optimization.* John Wiley, New York, second edition, 1987.

[13] W. Gander, G. H. Golub, and R. Strebel. Least squares fitting of circles and ellipses. *BIT*, 34:4:558–578, 1994.

[14] P. E. Gill, G. H. Golub, W. Murray, and M. Saunders. Methods for modifying matrix factorizations. *Math. Comp*, 28:505–535, 1974.

[15] Philip E. Gill and Walter Murray. Algorithms for the solution of the nonlinear least squares problem. *SIAM J. Numer. Anal.*, 15:977–992, 1978.

[16] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic Press, London, UK, 1981.

[17] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization. Volume 1.* Addison-Wesley, London, UK, 1991.

[18] G. H. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its application. *Inverse Problems*, 19:R1–R26, 2003.

[19] K. L. Hiebert. An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Trans. Math. Software*, 8:5–20, 1982.

[20] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[21] Linda Kaufman. Variable projection methods for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975.

[22] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003.

[23] H. P. Künzi, H. G. Tzschach, and C. A. Zehnder. *Numerical Methods of Mathematical Optimization.* Academic Press, New York, 1971.

[24] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[25] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications.* Wiley, New York, 1979.

[26] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[27] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM. Trans. Math. Software*, 5:64–85, 1979.

[28] J. J. Moré and S. J. Wright. *Optimization Software Guide.* SIAM, Philadelphia, PA, 1993.

[29] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[30] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces.* Academic Press, New York, third edition, 1973.

[31] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations.* Gordon and Breach, London, UK, 1970.

[32] P. Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations.* Gordon and Breach, London, UK, 1970.

[33] H. Schwetlick and V. Tiller. Numerical methods for estimating parameters in nonlinear models with errors in the variables. *Technometrics*, 27:17–24, 1985.

[34] S. J. Wright. *Primal-Dual Interior Point Methods.* SIAM, Philadelphia, PA, 1997.

Index